UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Diseño de circuitos de computación cuántica para cálculo de riesgos financieros mediante la aplicación de redes neuronales

Máster Universitario en Ingeniería de Telecomunicación

Autor: Raúl París Murillo
Tutor: Luís De Pedro Sánchez

FECHA: Febrero 2021

DISEÑO DE CIRCUITOS DE COMPUTACIÓN CUÁNTICA PARA CÁLCULO DE RIESGOS FINANCIEROS MEDIANTE LA APLICACIÓN DE REDES NEURONALES

AUTOR: Raúl París Murillo
TUTOR: Luís De Pedro Sánchez
PONENTE: Francisco Javier Gómez Arribas

High Performance Computing and Networking Research Group (HPCN)
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Febrero 2021

Resumen

Resumen

A lo largo de los últimos años se han producido grandes avances en el estado del arte de la computación cuántica, poco a poco las capacidades teóricas se están convirtiendo en realidad. Este cambio se está desarrollando al mismo tiempo que en el campo de la inteligencia artificial se está avanzando con pasos agigantados.

Estos avances ofrecen unas posibilidades ideales para abordar el reto del cálculo de los riesgos financieros. Actualmente, la cantidad de activos que manejan las entidades bancarias es demasiado grande para realizar la estimación del valor a riesgo (VaR) con los métodos clásicos, necesitando una capacidad computacional inmensa cuando el número de activos comienza a ascender. Además, en el cálculo del método más popular (Montecarlo) se utilizan aproximaciones a distribuciones gaussianas generadas con una serie de números aleatorios, que, al ser ejecutados en computadores clásicos no son totalmente aleatorios.

Para cubrir la necesidad de un nuevo método de cálculo del VaR surge este TFM. Así, en el mismo, se plantea desarrollar una implementación que realice la estimación del VaR de una forma completamente distinta, utilizando para ello una red neuronal que calculará las rotaciones de los qubits necesarias para crear un circuito cuántico que se ejecutará en un computador cuántico, para obetener el VaR.

El flujo de ejecución del diseño implentado comienza con la lectura de los activos introducidos, de los que se extraerá el VaR histórico, para realizar la comparacion en la red neuronal en el cálculo del coste de cada época. Estos datos iniciales serán divididos en entrenamiento y validación, para comprobar el correcto funcionamiento del sistema desarrollado.

Para llevar a cabo las pruebas necesarias, se ha desarrollado un sistema paralelo que no utilice el computador cuántico, de forma que se puedan comparar los resultados obtenidos entre ambos sistemas, y observar si la generación de números realmente aleatorios en el computador cuántico aporta beneficios. Además, se investigará la capacidad de la red neuronal para disminuir el efecto del ruido en los computadores cuánticos.

Las pruebas se realizan con los datos de tres y cinco activos, y se observarán distintas configuraciones de los parámetros personalizables del sistema desarrollado, para analizar el funcionamiento en detalle con las variaciones. Con la ejecución de este TFM, se ha comprobado que el sistema desarrollado es capaz de realizar el cálculo del VaR hallado un resultado cercano al valor del VaR paramétrico, situándose además, dentro del rango del VaR arrojado en distintas iteraciones del VaR de montecarlo.

Palabras Clave

Computación cuántica, riesgos financieros, redes neuronales, circuitos cuánticos, inteligencia artificial, qiskit, qubit.

Abstract

In the last few years, there has been important progress at the quantum computing state of the art, slowly, the theoretical capabilities have been transformed to the current reality. This change is being made while the artificial intelligence field is also progressing rapidly.

This progress offers an ideal possibilities to solve the current problem of the financial risks calculus. Nowadays, the amount of actives that are being managed by financial entities is too large to be able to estimate the VaR (Value at Risk) with classical methods, needing an immense computational capability when the amount of actives begin to increase. In addition, at the most popular estimation method (Monte carlo), there are used approximated gaussian distributions generated with series of random numbers. Those numbers are not totally random, because the method is executed in classic computers.

This end-of-master thesis develops from the need of finding a new method to estimate VaR. Also, it is planned to develop an implementation which calculate VaR from a different point of view, using for it a neural network that will calculate the needed qubit rotations to create a quantum circuit that will be executed at a quantum computer, obtaining the VaR.

The implemented design of the execution process is going to start with the read of the input actives, which will be used to extract historical VaR. This historical VaR will be used for comparison at the neural network cost calculus at each epoch. This initial inputs will be divided to training and validation, to check the correct functioning of the designed system.

In order to perform the needed tests, there will be developed another design that will not use the quantum computer to be able to compare both system results, and determine if the real random numbers generated by the designed system is beneficial or not. Also, there will be tested the capabilities of the neural networks to reduce the quantum noise effects.

The different tests will be done with data from three and five actives, which will be tested with different configurations of the developed system parameters to analyse the changes they produce. With the execution of this end-of-master thesis, it has been proved that the designed system is able to estimate the VaR with a similar obtained value to the parametric VaR, while also taking place into the range of the Montecarlo VaR estimations.

Key words

Quantum computing, financial risks, neural networks, quantum circuits, artificial intelligence, qiskit, qubit.

Agradecimientos

A mi familia, amigos y profesores.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

1.1. Motivación del proyecto

A lo largo los últimos años, la tecnología del campo de la computación cuántica[1] ha avanzado rápidamente y han alcanzadon algunos hitos que permiten su estudio e investigación. Lo que se planteaba como teoría hace años[2], se está convirtiendo en el estado del arte actual, eso sí, avanzando lentamente pero a un ritmo constante.

La teoría sostiene que es posible establecer una computación a nivel cuántico, estableciendo para el intercambio de información una versión cuántica de los bits clásicos, los qubits. Estos qubits realizarían la misma función en la computación que las que realizan los bits clásicos, siendo (en teoría) capaces de realizar operaciones entre ellos para obtener cálculos similares a las puertas lógicas clásicas.

Como es ampliamente conocido, una de las características limitantes de la computación clásica es no poder obtener números totalmente aleatorios, lo que podría producir un sesgo en algoritmos, generadores o modelos, tales como los algoritmos de Machine Learning[3] o Deep Learning, debido a que estos necesitan de una inicialización aleatoria para no caer en mínimos locales. Este problema de generar números totalmente aleatorios puede ser subsanado mediante la utilización de computadores cuánticos.

En los computadores cuánticos, mediante el uso de los qubits, existe la posibilidad de generar números completamente aleatorios debido a que los qubits tienen un resultado basado en su probabilidad[4], es decir, la ejecución de la misma operación puede dar resultados distintos. Esto no sucede en los computadores clásicos, dónde el resultado de operaciones idénticas es exactamente el mismo, cómo cabría esperar con un pensamiento axiomático, dónde la suma de dos más dos siempre da como resultado cuatro, por mucho que se realice la misma operación una y otra vez.

Pero el uso de computadores cuánticos para realizar cálculos tal y como lo hacen los computadores clásicos aún están en su fase inicial, puesto que no se tienen los conocimientos suficientes del funcionamiento real de estos sistemas, y se está investigando para alcanzar este hito en estos próximos años. La tecnología de los computadores cuánticos todavía es demasiado novedosa y se está desarrollando, dando lugar a la utilización de pocos qubits para estos sistemas, algo impensable en la computación clásica, donde el número de bits disponible para hacer

operaciones crece día a día a un ritmo vertiginoso. Es totalmente común encontrarse sistemas cuánticos de pocos qubits, tales como, cinco, seis, etc. De forma que tampoco se cumple la similitud en cuestión de ser un número al que se eleva el número dos, dando lugar a valores como dos, cuatro, ocho, etc.

En esencia, los qubits son estados cuánticos de un sistema al que se le puede aplicar cambios controlados (puertas cuánticas) para modificar su estado. Su representación necesita de matemáticas complejas y los cambios puede entenderse como rotaciones en un espacio de Hilbert multidimensional. Se puede establecer cierta analogía entre los niveles de voltaje que representa un bit en electrónica clásica y las puertas lógicas que permiten realizar la funcionalidad de un ordenar clásico, pero la manera de plantear algoritmos es radicalemente distinta. La descisiones sobre la utilización de distinta cantidad o tipo de puertas viene dada por el programador o diseñador del circuito cuántico, obteniéndose, tras la sucesión de las mismas, los resultados del circuito. De forma que se puede observar los estados binarios obtenidos, que varían en cada ejecución. Esta variación causada por la imposibilidad de saber si un qubit se encuentra en un 0 o un 1 con una exactitud del 100 % desencadena en que estos circuitos cuánticos necesiten de un número "n" de ejecuciones para sacar la distribución de probabilidades del resultado. De esta forma se habilita la observación de los mismos para que el usuario realice la elección (basándose en la distribución de probabilidad) de cuál es el resultado real obtenido, que en la mayor parte de los casos suele seleccionarse el resultado que más se ha obtenido de media.

Con la posibilidad de realizar operaciones de forma estadística debido a la aleatoriedad de los propios resultados de ejecución, los computadores cuánticos podrían mejorar la precisión en cálculos estadísticos, tales como el cálculo del VaR de activos bancarios. En este ámbito, el estado del arte de los computadores clásicos se encuentra en un punto en el que las operaciones clásicas son tan complejas y pesadas para los los mismos, que no son capaces de sacar el VaR por el método de Montecarlo[5][6] cuando se tiene una gran cantidad de activos correlacionados, por lo que tienen que emplear algoritmos aproximados, como, por ejemplo, el método paramétrico.

Principalmente, para el cálculo de riesgos financieros se utilizan actualmente dos métodos distintos, el método paramétrico y el método de Montecarlo. Al calcular el VaR mediante el método de Montecarlo se calcula un valor bastante cercano al riesgo real, pero su complejidad es tal que no se puede realizar este método cuando se tiene un número mediano de activos, debido a que la complejidad matemática es demasiado elevada para un computador clásico. De forma que cuando los activos correlacionados son un número elevado se emplea el método paramétrico, que reduce la complejidad matemática, pero aun empleándose este método aproximado, en la actualidad, hay tantos activos correlacionados para calcular los riesgos que no se suelen emplear su totalidad porque incluso este método es costoso.

Por esta razón, en este TFM se pretende adquirir el conocimiento necesario sobre los circuitos cuánticos para poder realizar los cálculos a través de los mismos[7], permitiendo de esta forma realizar cálculos mediante redes neuronales estableciendo las rotaciones de los qubits[8] necesarias para obtener una aproximación al método de Montecarlo y a la vez permitiendo utilizar para el cálculo todos los activos correlacionados, sin chocar con el máximo de la capacidad computacional de los cálculos matemáticos necesarios.

Si se tiene en cuenta esta aleatoriedad al realizar el cálculo, se podría mejorar la precisión al no tener que modelar la función estadística de resultado obtenida para el cálculo del VaR y la capacidad de aprendizaje y capacidad computacional que aporta una estructura de red neuronal[9][10], podría existir la posibilidad de realizar estos cálculos de una forma relativamente precisa y rápida, en esto se enfocará esencialmente el trabajo fin de máster. Además, se intentará realizar un diseño que sea capaz de funcionar en un computador cuántico real (con ruido), aprovechando las capacidades de aprendizaje de la red neuronal, dado que este ruido es el mayor problema de los diseños cuánticos actuales.

El desarrollo de la red neuronal se realizará intentando maximizar la precisión del cálculo del riesgo e intentando al mismo tiempo que la complejidad sea la mínima necesaria para que el tiempo en realizar el aprendizaje y el cálculo no sea demasiado elevado. Para poder implementar este desarrollo se utilizará el lenguaje de programación Python, al ser uno de los más populares en la actualidad y contener las librerías necesarias para la implementación del sistema cuántico (qiskit)[11][12].

En apartados posteriores se profundizará en el motivo de la elección de este lenguaje de programación para realizar la red neuronal, que, se pretende, que seleccione las rotaciones con precisión. Una vez calculadas las mismas el propio programa lanzará la ejecución en el computador cuántico, o el simulador cuántico en su defecto, para obtener el valor del VaR y calcular el riesgo de esta forma.

1.2. Objetivos y enfoque

Este trabajo fin de máster tiene como principal objetivo la profundización sobre el conocimiento actual de los computadores y circuitos cuánticos, mientras se desarrolla un sistema de aprendizaje automático basado en redes neuronales, para el cálculo del VaR en una serie de activos correlacionados.

Debido a la imposibilidad del cálculo por Montecarlo del VaR en computadores clásicos cuando el número de activos correlacionados aumenta en número, hecho que, en la actualidad, está sucediendo en el estado del arte, por lo que se aplican algoritmos aproximados, tales como, el paramétrico.

Una vez obtenido este objetivo, se pasaría a analizar y comprender los resultados obtenidos para intentar aportar un beneficio al estado del arte actual de los computadores cuánticos. Además, se podrán adquirir conocimientos sobre el diseño de los circuitos cuánticos, observando las distintas rotaciones que se aplican como resultado de la red neuronal diseñada.

Para facilitar la comprensión del trabajo realizado en el trabajo fin de máster, se van a plantear los objetivos generales del mismo, de forma cronológica, para observar paulatinamente los pasos necesarios hasta alcanzar el objetivo final:

- Aprendizaje sobre el estado del arte cuántico.
- Desarrollo de programas y circuitos cuánticos simples.
- Desarrollo y análisis de circuitos personalizados.
- Implementación de algoritmos de I.A. simples para la obtención de las rotaciones de los qubits.
- Implementación de redes neuronales para realizar cálculos del VaR.
- Desarrollo del programa de entrenamiento y predicción final.
- Comprensión y análisis de los resultados obtenidos con el sistema.
- Estudio de los conocimientos adquiridos en la realización de las tareas.

1.3. Fases de realización del proyecto

El planteamiento, desarrollo y resultado de cada una de las tareas está contenido en la memoria, permitiendo al lector observar todos los procedimientos y razonamientos seguidos en la realización de las mismas.

Para facilitar la comprensión del desarrollo del trabajo realizado, se ha decidido ilustrarlo con un diagrama de Gantt, de forma que, con ayuda de una tabla anexa, que enumera las tareas representadas se pueda observar con un solo vistazo la totalidad del mismo .

En la elaboración del diagrama de Gantt se ha decidido utilizar para el eje temporal las semanas en las que se ha llevado a cabo el trabajo desempeñado, utilizándose de esta forma el color verde para las tareas "generales" mientras que se ha utilizado el color azul para las subtareas realizadas dentro de cada una de las tareas "generales". La diferencia entre las subtareas marcadas con azul más claro reside en algo puramente visual, de forma que de un solo vistazo se puedan distinguir las subtareas, sin necesidad de realizar mentalmente una separación.

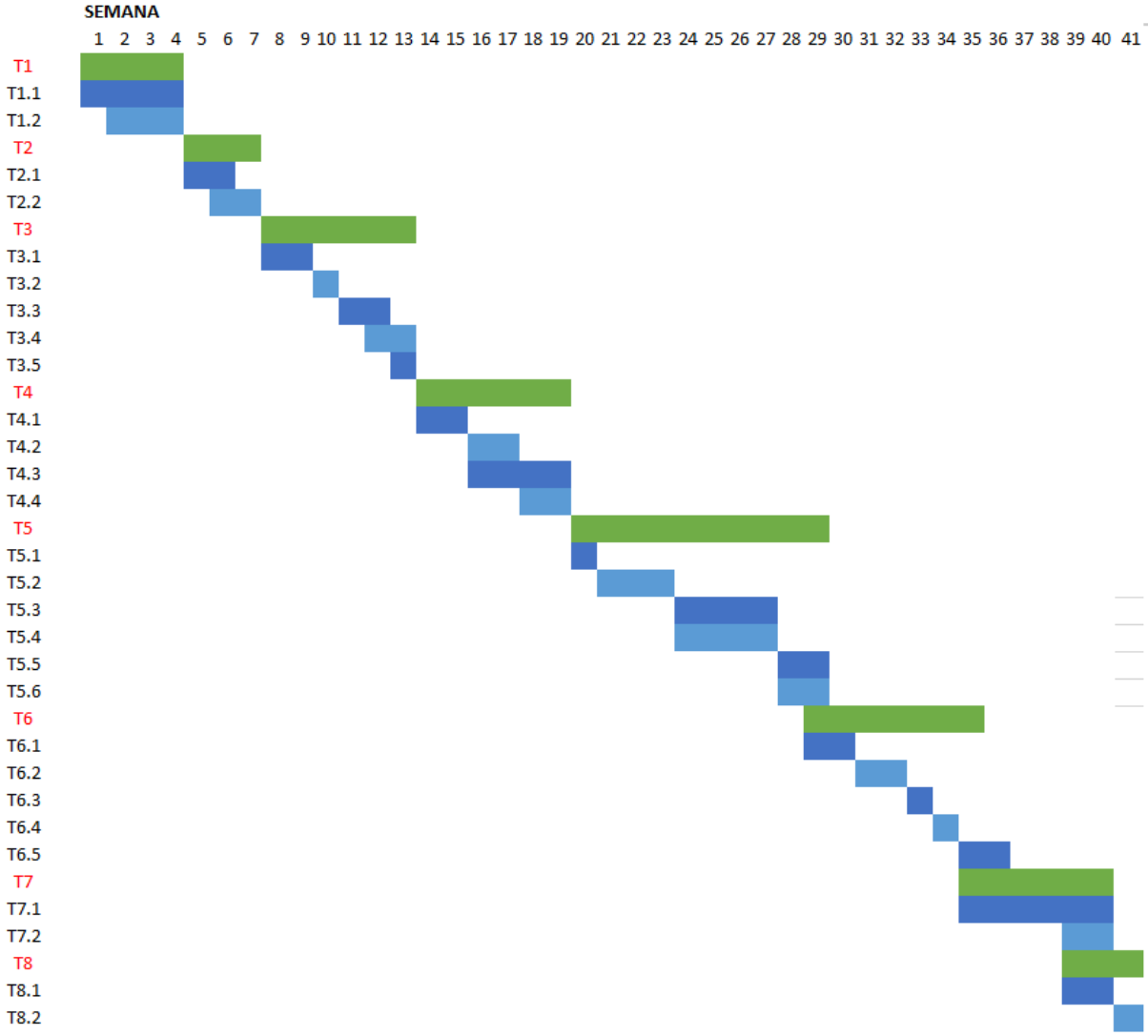

Figura 1.1: Diagrama de Gantt del proyecto

Una vez visto el desarrollo de trabajo, se puede visualizar la tabla con las tareas y subtareas realizadas para comprender qué trabajo se llevó a cabo en cada punto del desarrollo. De esta forma se puede ir al punto exacto en el que se ha realizado cualquiera de las tareas o subtareas que se han completado. De nuevo, para facilitar la lectura, se ha decidido resaltar las tareas "generales" con la letra en negrita.

Tareas / Subtareas	Horas
T1. Introducción al contexto cuántico	**30**
T1.1 Lectura de libros y artículos	20
T1.2 Lectura de la documentación de la librería Qiskit (python)	10
T2. Desarrollo de los primeros programas y circuitos	**15**
T2.1 Aprendizaje sobre los programas de ejemplo	5
T2.2 Desarrollo de programas simples con circuitos por defecto	10
T3. Desarrollo y análisis de circuitos personalizados	**44**
T3.1 Desarrollo de un editor de circuitos por defecto	15
T3.2 Prueba del correcto funcionamiento del editor	5
T3.3 Desarrollo de una función para extraer resultados	15
T3.4 Análisis de los resultados obtenidos variando las rotaciones	4
T3.5 Implementación de una nueva función para editar y ejecutar	5
T4. Implementación de algoritmos de I.A. simples para obtener el circuito ideal para el resultado deseado	**40**
T4.1 Implementación de un algoritmo de prueba aleatoria de rotaciones (fuerza bruta)	10
T4.2 Implementación de un algoritmo del tipo descenso de gradiente	15
T4.3 Aprendizaje sobre los resultados de las rotaciones obtenidas	5
T4.4 Prueba y adaptación de los algoritmos a distinto número de qubits	10
T5. Implementación de redes neuronales para realizar cálculos	**67**
T5.1 Desarrollo del esquema a implementar	5
T5.2 Desarrollo de la función de coste que recibe las rotaciones	20
T5.3 Implementación del entrenamiento de la red neuronal	15
T5.4 Desarrollo de una red neuronal saltándose el computador cuántico para comparar resultados	12
T5.5 Comparación de resultados obtenidos con ambos métodos	5
T5.6 Ajuste de los modelos para mejorar resultados	10
T6. Desarrollo programa de entrenamiento y predicción final	**50**
T6.1 Ajuste de los programas anteriores a número de qubits y datos variables	10
T6.2 Desarrollo de la función de entrenamiento y guardado del modelo resultante	15
T6.3 Desarrollo de la función de predicción de rotaciones con el modelo entrenado	5
T6.4 Realización de pruebas para el correcto funcionamiento	5
T6.5 Análisis de los resultados obtenidos y la precisión de la red neuronal desarrollada	15
T7. Realización de la memoria del TFM	**50**
T7.1 Redacción de la memoria	40
T7.2 Corrección de la memoria	10
T8. Defensa del TFM	**4**
T8.1 Preparación de la presentación	3
T8.2 Presentación del TFM	1

Cuadro 1.1: Tareas realizadas.

1.4. Organización de la memoria

En este apartado se va a explicar la división realizada en la memoria para el trabajo realizado durante el proceso del TFM.

Para facilitar su comprensión, se va a optar por realizar una enumeración de cada uno de los capítulos contenidos en la memoria, de forma que se vaya explicando capítulo a capítulo el contenido del mismo.

Cada capítulo necesitará de la lectura de los anteriores para su debida comprensión, aunque, en la medida de lo posible, se intentará facilitar el aprendizaje de cada uno de los apartados, para poder asimilarlos de forma correcta, independientemente de poseer, o no, los conocimientos de los capítulos anteriores.

- Introducción:

  En este capítulo se exponen las ideas iniciales para la realización del TFM, permitiendo de esta forma comprender los problemas que se pretenden solucionar. También en este apartado se plantean las tareas que se han realizado a lo largo del desarrollo del TFM otorgando una visión global del mismo. Para facilitar la visualización de las tareas y subtareas realizadas, se presenta un diagrama de Gantt acompañado del cuadro de tareas, con las horas desempeñadas en cada una de ellas.

  Una vez expuestas las ideas de las que se parte para el TFM, los objetivos planteados con sus tareas asignadas y duración de las mismas, se llega a esta parte, dónde se exponen las divisiones realizadas en la memoria para la correcta explicación de todas las acciones realizadas y el conocimiento adquirido con las mismas.

- Estado del arte:

  En este capítulo se expone el estado del arte actual de las tecnologías empleadas en cada una de las tareas del TFM. Para poder comprender los siguientes capítulos de la memoria y lograr entender el sentido del flujo de tareas realizadas, se necesita poseer una base sobre los conocimientos actuales de la computación cuántica y el cálculo de riesgos financieros. Debido a estas razones, este capítulo pretende otorgar al lector una serie de conocimientos mínimos sobre las bases de la computación cuántica, que sirvan para comprender el diseño de circuitos cuánticos, que, como ya se ha resaltado en la introducción, es una tecnología novedosa de la que todavía no se tiene un amplio conocimiento.

  De igual forma, para comprender el objetivo final del TFM, se necesita poseer unos conocimientos sobre la necesidad del cálculo de los riesgos financieros y los límites por los que no se pueden realizar cálculos clásicos cuando el número de activos correlacionados aumenta, situación que ya se está produciendo en la actualidad.

- Diseño:

  En este capítulo se exponen las decisiones iniciales tomadas para llevar a cabo el desarrollo de las tareas designadas. Antes de comenzar a desarrollar programas para completar cada uno de los pasos para alcanzar el desarrollo final, se tomaron bastantes decisiones de diseño que se han seguido en cada uno de los pasos del desarrollo. Este capítulo sirve de base para comprender los pasos llevados a cabo en el desarrollo, obteniendo una visión global de los mismos.

  También en este capítulo se explicarán las decisiones sobre dónde acometer los pasos del desarrollo, así cómo la explicación del motivo por el que todos los desarrollos se realizan sobre la librería de programación Qiskit, utilizando para ello el lenguaje de programación Python.

- Desarrollo:

  En este capítulo se exponen todos los procesos seguidos para la realización de cada uno de los pasos, explicando detalladamente las dificultades encontradas en cada uno de ellos y las decisiones que se han ido tomando a lo largo de su desarrollo. Dada la cantidad de tareas realizadas, se intentará, en la medida de lo posible, que cada una de ellas se

comprenda de forma individual, permitiendo al lector enfocarse en las tareas que considere de mayor importancia.

Se pretende seguir un orden cronológico a lo largo del capítulo para habilitar la capacidad de detenerse en la lectura, volver al diagrama de Gantt y observar en qué momento del TFM se encuentra el desarrollo expuesto. Una vez finalizada la explicación del desarrollo de cada una de las tareas, se expondrán los conocimientos adquiridos en la realización de las mismas.

- Implementación:

  En este capítulo se expone el diseño y desarrollo del programa final. Para poder alcanzar este punto, es necesario completar las demás tareas planteadas, de forma que el programa vaya haciéndose cada vez más complejo y pasando de un programa simple a un programa capaz de ejecutarse por sí solo, leyendo los datos, diseñando el circuito que variará las rotaciones de los qubits, la red neuronal que calculará dichas rotaciones y extrayendo los resultados obtenidos.

  Se pretende que este programa también posea la capacidad de adaptarse a distinto número de activos automáticamente, que el programador pueda cambiar el número de qubits utilizado de forma sencilla una variable constante, que se pueda ajustar el modelo de la red neuronal a elección del programador, además de una serie de variables de diseño que serán ajustadas como constantes.

- Resultados:

  En este capítulo se exponen las pruebas realizadas con el programa final y los resultados obtenidos con las mismas. Así mismo se realizará un examen del efecto de la red neuronal aplicada a los circuitos cuánticos diseñados, y las diferencias que se encuentran al realizar la ejecución en el sistema cuántico simulado y el real.

  Una de las principales diferencias entre la simulación y el real, reside en que el sistema real introduce un ruido en las ejecuciones del circuito[13], haciendo que los resultados sean menos exactos. Se profundizará en esta diferencia en el capítulo correspondiente.

- Conclusiones y trabajo futuro:

  En este capítulo se exponen las conclusiones obtenidas del TFM realizado, comparando las propuestas inicialmente planteadas con el trabajo real realizado, y observando si los resultados de los experimentos han sido satisfactorios, o por el contrario, no se ha logrado el objetivo deseado. Con todos estos resultados se planteará el conocimiento adquirido a lo largo de la realización de todas las tareas y experimentos, proponiendo una serie de trabajos futuros que podrían derivarse de este TFM, o que podrían hacer uso de los conocimientos adquiridos en el mismo.

2
Estado del arte

En este capítulo se va a pasar a exponer el estado del arte actual de las tecnologías empleadas en cada uno de los pasos realizados a lo largo del TFM. Se han divido los conocimientos en grandes apartados para poder diferenciar cada una de las tecnologías de las demás.

Se va a intentar expresar todos los conceptos de la forma más sencilla posible, para habilitar la comprensión de la mayoría de las personas (con una base previa), de esta forma, se pretende lograr que conceptos complejos queden claros para el lector.

2.1. Computación cuántica

Como se ha comentado en la introducción, la computación cuántica es una tecnología que está en pleno auge en los últimos años. Comenzó siendo planteada teóricamente y sin tener una fecha prevista para que la teoría se hiciese realidad, en gran parte debido a su gran complejidad y la imposibilidad que existía con el estado del arte. Pero con el paso del tiempo, se ha podido observar que esta tecnología se está desarrollando lentamente, convirtiendo poco a poco la teoría en realidad.

2.1.1. Capacidad teórica

Para poder comprender el beneficio que la computación cuántica podría aportar a las personas, primero hay que entenderla de forma básica. No sirve solo conocer todas las nuevas posibilidades que se abrirían a la humanidad, sino también, conocer las bases de la computación cuántica y el motivo de esa mejora.

En este apartado se considera que la manera más sencilla de comprender un concepto tan complejo, es partir de axiomas conocidos para la mayoría de los lectores e ir abstrayendo hasta lograr comprender la computación cuántica de forma básica. No es necesario ser un experto en computación cuántica para comprender este TFM, pero sí son necesarios unos conceptos mínimos.

Si se parte desde lo más básico, la computación clásica podría ser explicada como: "Un conjunto de corrientes eléctricas que logran realizar una operación lógica". Usando como

operaciones lógicas las archiconocidas operaciones: "OR", "AND", "XOR", "NOT", así como sus variantes negadas. De esta forma, se puede comprender que cualquier operación que se realice en un computador clásico, necesitará forzosamente una corriente eléctrica, siendo su unidad atómica, los "bits".

Probablemente, hayan venido a la mente algunas acciones que están dentro de la computación clásica que podrían no utilizar necesariamente operaciones lógicas directamente, como almacenar una variable en un buffer, pero indirectamente, hay un multiplexor o un "flip-flop" que está habilitando el almacenamiento mediante alguna operación lógica. Si se habla con propiedad, siempre se pueden encontrar transistores.

Partiendo de que ya se tiene una idea de qué es la computación clásica, se puede empezar el proceso de abstracción. En la computación clásica, se utiliza para realizar las operaciones lógicas las principales características de la electricidad, es decir, la intensidad, la resistencia y la tensión. De forma que, aunque se estén utilizando electrones en el proceso, dato importante para la computación cuántica, no se emplea la faceta cuántica que podría poseer en superposición cuántica, sino se emplea al electrón como un conjunto que forma el fenómeno físico que se conoce como corriente eléctrica.

La corriente eléctrica, como es conocido, se podría definir como el proceso físico que se produce con el desplazamiento de los electrones y las fuerzas que ellos mismos producen a lo largo del recorrido. Si observamos un cable de cobre como una serie de fuerzas que atraen a los electrones desde el punto con más resistencia hasta el mínimo, se puede hacer un símil con una tubería de agua, donde el agua tiende también a ir al punto de mínima energía. De modo que muchos conceptos eléctricos se pueden definir con conceptos conocidos por cualquier persona, como una cascada de agua para definir el voltaje o una presa en un rio para la resistencia.

Teniendo estos conceptos en mente, se pueden presentar otros elementos básicos relacionados con la corriente eléctrica, las resistencias. Estos elementos podrían resultar a veces anti intuitivos para el lector, porque, ¿para qué se va a querer poner resistencia en un circuito? ¿No sería mejor reducir la resistencia al mínimo? En efecto, la respuesta es simple, sí son necesarias resistencias en la gran mayoría de circuitos eléctricos. Esto se debe a que todos los componentes electrónicos, como las puertas lógicas, necesitan de un rango de tensión e intensidad, pudiendo fundirse si estos límites se sobrepasan o no funcionar de forma óptima si se reducen demasiado. En esencia, las resistencias son la forma de "regular" la corriente y la tensión, sirviendo de limitantes en los cables y de guía para el circuito, afectando de igual forma que la cantidad de carriles en una carretera.

Una vez se conoce la intensidad, la tensión y la resistencia, se puede abstraer la idea un poco más para preguntarse: ¿Por qué la computación clásica es más lenta que la computación cuántica? Para comparar dos conceptos, es condición necesaria tener conocimientos de ambos. Primero se va a pasar a explicar el proceso que sigue la computación clásica para obtener un resultado de las operaciones, pudiendo al final, apreciar esta "lentitud teórica" frente a la computación cuántica.

Las operaciones lógicas en la computación clásica necesitan de varias acciones eléctricas para poder realizarse. Se intentará explicar este proceso de forma cronológica para que se aprecie. En efecto, el tiempo necesario para la ejecución de las mismas es un tiempo imperceptible para los humanos, pero que sumados unos con otros pueden hacer que un proceso computacional sea "lento" a ojos computacionales. Cabe destacar que este tiempo "lento" se trata de microsegundos, es decir 10^{-6} segundos.

Primero, las operaciones lógicas necesitan rellenar los bits de la puerta de entrada, es decir, es necesario que si el bit se encuentra a "1", la tensión o la corriente de esa pata de la puerta de entrada se encuentre activa, mientras que si el bit se encuentra a "0", esa pata necesita vaciarse de corriente y de tensión para lograr estar en ese estado. Este proceso de rellenado de bits a la entrada de la puerta se podría considerar casi instantáneo por el bajo tiempo que necesita,

pero no puede ser despreciable, ya que incluso en la mayoría de los sistemas actuales ya se tiene en cuenta el tiempo que tarda la corriente en desplazarse por el cable. Debido a que el cable también tiene una resistencia intrínseca (por no ser un material superconductor). Por lo que ya tenemos el primer incremento o delta de tiempo necesario, desde el inicio de la operación hasta que los bits están presentes para la operación en la puerta de entrada.

Una vez tenemos los bits cargados en la puerta de entrada, estos entran a través de la misma y realizan las distintas operaciones físicas que componen la puerta. Cada una de estas operaciones necesita un tiempo concreto. En las puertas lógicas se suele hablar del tiempo necesario para que el resultado se encuentre en la pata de salida, es decir, que ya se hayan realizado las operaciones necesarias y se encuentre el resultado disponible para el siguiente proceso.

Teniendo el resultado en la pata de salida, todo lo que queda es que este resultado llegue al punto deseado final, que puede ser a otra puerta lógica o cualquier otro lugar para que la operación se complete correctamente.

Conociendo el proceso de ejecución de una puerta lógica, queda añadir una variable que afecta a todo el circuito, esta es, una señal eléctrica que se llama "de reloj", ampliamente conocida como "CLK". Esta señal afecta a todo el circuito de forma que no se pueden realizar operaciones síncronas (afectadas por el reloj), sin que esta señal de paso, por lo que, aparte del tiempo necesario para el cálculo del resultado, también se necesita que el reloj de paso a las operaciones (excepto si son asíncronas).

Este proceso de control mediante una señal eléctrica se debe a que, de esta forma, se puede serializar distintas operaciones lógicas, pudiendo empezar a cargar los bits para otra operación cuando una acaba de llegar a la pata de salida. Para ello se utilizan una serie de flip-flops para ir almacenando los valores binarios de los bits. Pero esto produce también que muchas operaciones tengan que ir al tiempo en el que va la "batuta del director", teniendo que esperar, aunque termine antes a la siguiente señal.

Para intentar evitar este efecto, se realizan muchas operaciones de forma paralela, cuántas más operaciones se puedan hacer paralelamente, mejor. Si se realizan paralelamente, se evita tener que esperar que otra operación termine previamente para poder empezar a realizar la siguiente, por lo que es un trabajo muy importante para el procesador, el intentar paralelizar los procesos al máximo.

Aun intentando utilizar todas las posibilidades para aumentar la paralelización o aumentar el ritmo de la señal de reloj para que las operaciones se realicen más rápidamente, se puede observar la barrera física que se encuentra, el tiempo que se ha presentado de cargar bits, realizar las operaciones físicas y enviar el bit al siguiente punto, casi no se podría mejorar (en comparación con las velocidades cuánticas), incluso en algún caso, no se puede mejorar nada (superconductores).

Por lo que encontramos una barrera en la computación clásica, desde sus bases físicas, hasta los procesos realizados en el computador clásico actual, dónde existen una serie de tiempos que incluso aumentando la velocidad de los desplazamientos de los cables hasta el infinito (algo imposible de realizar), existiría un límite del orden de los microsegundos actualmente, en el mejor de los casos, para realizar una operación lógica clásica.

Pero se ha comentado que las velocidades son prácticamente instantáneas al ojo humano y muy rápidas desde un punto de vista computacional, ¿Cómo puede ser que la computación cuántica produzca tales beneficios, que se considere no un paso adelante, sino, un salto hacia la nueva era computacional?

Pues resulta ser una respuesta concisa, pero para nada sencilla, utilizar para el cálculo computacional unas de las propiedades que más rápido se realizan en el universo (conocido), la superposición estados cuánticos y el entrelazamiento.

Se podría realizar una comparación con las redes eléctricas que proporcionan ancho de banda a las viviendas actualmente, hace años, todos los cables eran de cobre y se podía alcanzar unas velocidades que, finalmente, estaban alcanzando límites en los que otorgar un ancho de banda muy superior era algo muy complejo y costoso para la proveedora. Pero, de pronto, apareció un nuevo factor para proveer el servicio, la fibra óptica. Este nuevo cable es capaz de proporcionar velocidades inalcanzables para los cables de cobre convencionales y ofrecer al consumidor un nuevo ancho de banda que no podía imaginar hace años.

La principal diferencia entre la computación cuántica y la computación clásica, es el elemento atómico que utilizan para realizar las distintas operaciones lógicas, los qubits frente a los bits clásicos. Esta diferencia produce que los tiempos de estas operaciones y su complejidad física sea muy lejana una de otra. En la siguiente sección se abordará con mayor profundidad la diferencia entre estos elementos de la computación, que por ahora tomaremos como simples contenedores de información que se utilizan para realizar las operaciones y obtener el resultado.

Como era previsible, las puertas lógicas clásicas no sirven para realizar operaciones con computación cuántica. Debido a esto, se encuentra el principal problema de la computación cuántica, ¿Cómo se puede hacer una puerta lógica para estados cuánticos? Poco a poco se van respondiendo a esta serie de preguntas con mayor claridad, aunque cabe destacar que, por ahora, los sistemas cuánticos están en nacimiento, mientras que los sistemas clásicos son ampliamente conocidos desde el punto de vista físico, y dónde la parte clásica no sorprende, los sistemas cuánticos no dejan de sorprender a los investigadores.

Si se logra resolver el problema de realizar puertas lógicas que funcionen correctamente para la computación cuántica, al igual que ocurre en la computación clásica, se encuentra que el tiempo que estas operaciones necesitan sería muy inferior, debido a que se estarían realizando cálculo y transporte de información a valocidades muy superiores. Por lo que se tendría que, cálculos que necesitan de decenas de miles de años en computación clásica, se pueden realizar en cientos de segundos en computación cuántica, dando un salto hacia delante abrumador en el campo de la computación. Causando la necesidad de replantear grandes axiomas de la computación clásica, como por ejemplo, la nueva necesidad que existiría de crear un sistema de claves, puesto que, se podría llegar a realizar ataques de fuerza bruta a claves que actualmente se consideran "indescifrables", dado el tiempo que necesitaría el mejor de los sistemas clásicos en descifrarla.

2.1.2. Qubits

En esta subsección se va a profundizar en la diferencia existente entre los bits clásicos y los qubits, pudiendo comprenderse de esta forma las cualidades que hacen de los qubits unos perfectos portadores de información en la actualidad.

Los qubits, o átomos de información de los circuitos cuánticos, son estados cuánticos "controlados", es decir, que se posee un estado de superposición controlado y se puede comprobar el estado cuántico en el que se encuentran. Mientras que en la computación clásica los valores en los que estaba un bit venían definido por su tensión o intensidad, en el caso de los qubits viene definido por la posición o estado en el que se encuentran en el momento de ser medidos.

Por una parte, los bits clásicos tienen dos estados posibles 0 y 1. Estos estados vienen definidos por el diseñador, no teniendo por qué ser exactamente estos valores, de hecho, en algunos sistemas se sitúa el 0 binario en el mismo valor absoluto que el 1 binario, pero con signo negativo. De esta forma se puede reducir el efecto del ruido en el sistema o ayudar a los decisores a elegir uno y otro.

Sin embargo, aunque los qubits también tengan dos estados que serían considerados como 0 y 1, existe en ellos una característica que los diferencia ampliamente de sus predecesores, esta característica es la propiedad de superposición cuántica.

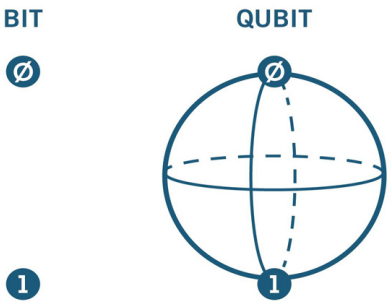

Figura 2.1: Representación de un bit clásico y un qubit[14]

La superposición cuántica es la capacidad que tienen los qubits de poseer simultáneamente ambos estados. Para lograr esta superposición se debe manipular el objetivo, para alcanzar este estado cuántico con láseres de precisión o microondas de fases.

Observando la representación de los qubits se puede apreciar que los estados de un bit clásico tienen solo una dimensión, puede estar situado del lado del 0 o del 1. Mientras que si se aprecia la representación del qubit se puede observar que el espacio de estados pasa a ser bidimensional (un plano), plegado alrededor de la esfera formada por el qubit, puesto que para que el valor sea 0 no se debe estar necesariamente en el punto exacto, basta con estar en el hemisferio norte del qubit, al igual que con su opuesto, dónde bastaría con encontrarse en el hemisferio sur del mismo.

Este fenómeno de la superposición cuántica es aprovechado para realizar operaciones con los qubits, que de hecho se podrían definir como: un sistema cuántico que se encuentra en superposición de todos sus posibles estados. Esto se produce debido a que la posición real de un qubit es probabilística, como se ha explicado en la computación cuántica y lo cuál es el motivo de la necesidad de un número suficiente de ejecuciones para obtener un resultado aceptable.

Que un qubit se encuentre en un estado u otro al ser medido, depende de la probabilidad que tengan esos estados, dando por ende, un resultado probabilístico. El fenómeno de determinar si un qubit se encuentra en un estado o en el contrario se denomina: medición cuántica. En este proceso físico se determina en qué estado se encuentra, que como ya se ha comentado, puede ser cualquiera de los dos estados, incluso en regiones entre medias de ambos en cualquier punto bidimensional de la esfera, por lo que se podría definir al punto que se encuentra en la esfera como un número con dos variables, es decir, un número complejo, mientras que los bits clásicos siempre se encontrarían como números binarios.

Durante el proceso de medición cuántica se observa el estado en el que se encuentra el qubit en ese instante, descartando uno de los dos estados, y otorgando de esta forma un resultado que se puede utilizar en los subsiguientes cálculos o como resultado final de una operación.

Esta superposición de estados, que se podría comparar con la paradoja del gato de Schrödinger, es un estado de los qubits muy frágil, pudiendo deshacerse en cualquier momento si no se tiene un cuidado especial con el qubit. El proceso por el que un qubit deja de estar en el estado de superposición se llama: decoherencia cuántica.

La decoherencia cuántica consiste en la caída del qubit desde la superposición, a un estado en concreto, puesto la superposición cuántica es un estado sumamente delicado e inestable en un sistema, la decoherencia es un proceso que se espera que suceda, aunque no es demasiado predecible. Para aclarar este proceso, se podría comparar con la apertura de la caja en el caso del gato de Schrödinger, dónde finalmente se descubre si el gato sigue vivo o, en cambio, se envenenó y murió.

Muchos de los esfuerzos para que la computación cuántica pueda suceder residen en intentar que el proceso de decoherencia cuántica no suceda a los qubits, dado que, si esto ocurriese, se necesitaría volver a colocarlo en un estado de superposición cuántica o de seleccionar otro objetivo para ser qubit, para volver realizar el proceso desde el inicio. La forma en la que esto se realiza en la actualidad es diversa, pero todos los métodos intentan evitar que lleguen al qubit variaciones de temperatura o vibraciones. Los métodos más comunes protegen al qubit del mundo exterior situando al qubit casi a 0 K, o situando al qubit en el interior de una potente cámara de vacío. De esta forma se intenta evitar la pérdida de la superposición.

Cómo es habitual en las comunicaciones, y con los qubits no iba a ser menos, también existe el pensamiento de colocar en los sistemas qubits extras de seguridad, es decir, réplicas de seguridad que pueden sustituir a cualquiera de los qubits en caso de que alguno de estos pierda su estado de superposición. Sin embargo, esto es un proceso costoso, dada la gran energía que se necesita para mantener cada uno de los qubits en su estado superpuesto.

Pero cómo suele ocurrir, esta protección es una relación de "protección-ruido", dónde a la vez de proteger el qubit de la decoherencia cuántica, se produce un ruido capaz de afectar al correcto funcionamiento de los qubits, además del ruido del exterior del que se está intentando proteger. Debido a esto, los sistemas cuánticos actuales tienen un ruido inherente que se debe intentar reducir con los cálculos. Por ejemplo, los algoritmos cuánticos con inteligencia artificial podrían reducir el efecto del ruido en las operaciones con los qubits, aunque este proceso de reducción está en una fase muy temprana todavía, y no se realiza de forma óptima.

Por otra parte, los qubits tienen otra propiedad adicional que resulta útil a la hora de utilizarlos para realizar operaciones con ellos, esta propiedad es: el entrelazamiento. En entrelazamiento de los qubits es, en esencia, la "unión" de dos qubits distintos de forma que pasen a estar correlacionados y, midiendo uno de ellos, se puede saber el estado en el que esta el otro.

Es decir, en este estado, ambos qubits pueden alterar el uno el estado del otro y se encuentran compartiendo el mismo estado en todo momento (a la hora de la medición cuántica), pudiendo utilizarse para comunicar cálculos u operaciones a otro punto elegido con el otro qubit del par de qubit entrelazados cuánticamente. Esta propiedad resulta abrumadora cuándo se descubre que este entrelazamiento no se rompe con la distancia, se pueden separar los qubits cualquier distancia, y estos comunicarán los cambios en el estado al otro instantáneamente para que en el momento de la medición ambos presenten el mismo estado cuántico, 0 o 1.

El entrelazamiento cuántico se puede lograr de forma empírica, pero no se tienen conocimientos, en la actualidad, del procedimiento por el que estos qubits quedan unidos en un solo estado cuántico. Tal es el vacío de la teoría física, que resulta un proceso casi incomprensible actualmente, y al que el mismísimo Einstein describió como: "una acción remota aterradora". Ya que no se comprende a qué se debe la unión, ni que fuerza es la que realmente enlaza los qubits para mantenerse siempre en el mismo estado, sin importar la distancia a la que están separados.

El entrelazamiento es utilizado en los computadores cuánticos para realizar cálculos a una velocidad titánica, y mediante ella ocurre un fenómeno que separa la capacidad computacional clásica de la capacidad cuántica. Al introducir un nuevo bit clásico en un sistema clásico, este tiene una capacidad de procesamiento (teórica) del doble de su capacidad anterior, sin embargo, al introducir un nuevo qubit en un sistema cuántico la capacidad del nuevo sistema no se duplica, sino que crece exponencialmente. Este fenómeno produce que unos pocos qubits puedan tener más capacidad computacional que un computador clásico potente.

Una comparación que se suele utilizar en el ámbito de la computación cuántica es el de decir que la capacidad computacional ideal se producirá cuando se tengan cien qubits óptimos que puedan realizar operaciones tal y como lo realizan los computadores clásicos, puesto que simplemente con cien qubits la capacidad supera de tal forma a la computación clásica, que esta

quedaría totalmente obsoleta. Se debe destacar que, estos "simples cien qubits", quedan todavía en un horizonte lejano, ya que, aunque la afirmación sea concisa, no resultan tan simples como parecen.

Pero, habiendo visto todas las posibilidades que otorga la computación cuántica y sabiendo que se está investigando con todos los esfuerzos sobre ella, ¿En qué punto se encuentra la computación cuántica actualmente? ¿Los sistemas actuales ya se acercan al punto en el que se encontraría la supremacía cuántica?

2.1.3. Sistemas cuánticos actuales

En este subapartado se va a exponer la situación en la que se encuentran actualmente los sistemas cuánticos, para otorgar así, una visión de la capacidad que tendrían los mismos.

Se podría definir un sistema cuántico como un conjunto de qubits capaces de realizar acciones coordinadas para obtener cálculos computacionales. Sin embargo, los circuitos cuánticos no son exactamente como los circuitos clásicos, dónde se habilitan ciertas puertas lógicas y cierran otras para realizar operaciones, pero eso se verá en el siguiente apartado de la memoria, por ahora este apartado se centrará en el sistema que habilita el posterior diseño de circuitos en el mismo.

En la actualidad existen algunas compañías que permiten la utilización de sus sistemas cuánticos propios en remoto, de forma que un usuario puede realizar peticiones de ejecución para que el circuito o proceso cuántico se ejecute y le sea mostrado el resultado de esta ejecución.

Para la investigación esta cadena de ejecución dónde no es necesario poseer una computadora cuántica, sino que se puede utilizar directamente en la nube de los proveedores, es muy beneficioso, puesto que permite desarrollar y ejecutar circuitos cuánticos a cualquier investigador con interés en ello. Esto supone que la tecnología de la computación cuántica pueda dar pasos agigantados en cortos espacios de tiempo, acercando cada vez más el horizonte cuántico al presente.

Un ejemplo de proveedoras de computadoras cuánticas en la nube podrían ser: IBM[15], Rigetti, D-Wave o incluso Alibaba. Estos proveedores suelen suministrar unos sistemas cuánticos gratuitos para todos los usuarios, y además, una serie de sistemas cuánticos potentes para usuarios identificados. Para ilustrar cómo sería la visión de estos sistemas cuánticos de una de estas proveedoras, se va a mostrar los sistemas cuánticos disponibles en IBM, ya que estos sistemas serán utilizados a lo largo del TFM.

Name	Qubits	QV	Status	Total pending jobs	Processor type	Features
ibmq_santiago	5	32	Online	39	Canary r3	-
ibmq_valencia	5	16	Online	454	Canary r2	-
ibmq_vigo	5	16	Online	1290	Canary r2	-
ibmq_16_melbourne	15	8	Online	1210	-	-
ibmq_ourense	5	8	Online	1539	Canary r2	-
ibmq_5_yorktown	5	8	Online	460	-	-
ibmq_qasm_simulator	32	-	Online	44	-	-
ibmq_athens	5	32	Paused - In use	27	Canary r3	-
ibmq_armonk	1	-	Maintenance	7	Emu r1	

Figura 2.2: Sistemas cuánticos disponibles en IBM

Como se puede apreciar, en este ejemplo hay nueve sistemas disponibles para su utilización. Si se pasa a observar brevemente cada una de sus columnas y el significado de las mismas se puede apreciar que la tabla comienza con el nombre de cada uno de los sistemas. Este nombre contiene un prefijo que siempre se mantiene, donde se nombra al proveedor y se indica con la letra "q" que se trata de un sistema cuántico. Posteriormente, se encuentra el nombre del sistema. Estos sistemas están distribuidos a lo largo del globo terrestre, intentando que siempre haya uno cerca de los usuarios y colocándolos preferiblemente en zonas con baja actividad sísmica, para que los qubits reciban una menor cantidad de ruido por la vibración terrestre.

En la segunda columna se muestran la cantidad de qubits que contiene cada uno de los sistemas, pudiendo apreciarse que generalmente estos sistemas poseen cinco qubits, y que el sistema de dieciséis qubits es un caso excepcional. También se puede ver que existe un sistema de treinta y dos qubits disponibles, este sistema es un simulador cuántico que realiza los cálculos aproximadamente cómo lo haría un sistema cuántico con esa cantidad de qubits, aunque, al ser un simulador, este sistema no introduce ruido en los cálculos, y no se puede asegurar que se vayan a producir los mismos resultados en un sistema cuántico real de la misma cantidad de qubits.

La tercera columna es un dato bastante significativo, se trata del volumen cuántico. El volumen cuántico es una medida creada por IBM para poder determinar el rendimiento de un computador cuántico en base a sus capacidades y su tasa de error. Al depender el correcto funcionamiento de un sistema cuántico de tantas variables distintas, se ha creído conveniente crear una nueva medida para estos sistemas, otorgando así una idea de las capacidades de procesamiento de ese sistema. En general, cuanto mayor sea el valor del volumen cuántico de un sistema, mejor rendimiento tendrá el mismo.

En la cuarta columna se aprecia un valor ampliamente conocido en sistemas en la nube, si el sistema está disponible para recibir trabajos y ejecutarlos, o en cambio está en mantenimiento o pausado. Aunque esta columna parezca arbitraria desde el punto de vista de un lector acostumbrado a proveedores con una amplia disponibilidad, en estos sistemas cuánticos se realizan a menudo ajustes y calibraciones para que funcionen correctamente, por lo que a la hora de mandar un circuito a ejecutar hay que consultar primero que sistemas están disponibles actualmente.

La quinta columna no requiere de una amplia explicación, simplemente se puede encontrar la

cantidad de procesos encolados para su ejecución. Suele ser una buena práctica intentar enviar los procesos a sistemas que tengan menor número de trabajos encolados, ya que, en ocasiones, la espera hasta que se realiza un proceso puede llegar a tardar decenas de minutos. Y como se puede suponer, la espera para la ejecución siempre será mucho más larga que la ejecución del proceso, puesto que estamos tratando con sistemas cuánticos.

Por último, encontramos las columnas del tipo de procesador y las características adicionales de cada sistema. No se pretende profundizar en los procesadores de los sistemas cuánticos, pero, en concordancia con las notaciones en los sistemas clásicos, cuanto mayor sea el número de la versión, mejor debería ser el procesador. Las características adicionales se muestran en la última columna, en este caso, solo el último sistema tiene una característica, que en este caso es: soportar pulsos eléctricos como input.

Toda esta información mostrada en la tabla resumida es útil en el momento de elegir un sistema cuántico para la ejecución, pero en el caso de que se desee profundizar más en cada sistema y realizar una elección más detenida, IBM también proporciona información extra sobre cada uno de estos sistemas. Por ejemplo, se ha decidido mostrar los últimos valores de la última calibración y las características de ejecución que posee la computadora cuántica de Santiago de Compostela, que cómo se podía apreciar era el sistema de cinco qubits más "fiable".

Topology & calibration data — Last calibrated: 12 hours ago

Topology diagram | Individual qubit properties | **Calibration data**

Qubit	Frequency (GHz)	T1 (µs)	T2 (µs)	Readout error	Sqrt-x (sx) error	CNOT error
Q0	4.833	54.53	103.99	1.750e-2	3.816e-4	cx0_1: 1.000e+0
Q1	4.624	192.46	94.37	1.600e-2	1.540e-4	cx1_2: 1.000e+0 cx1_0: 1.000e+0
Q2	4.821	148.87	64.04	1.250e-2	1.827e-4	cx2_3: 6.419e-3 cx2_1: 1.000e+0
Q3	4.742	183.2	98.27	3.070e-2	2.256e-4	cx3_4: 7.193e-3 cx3_2: 6.419e-3
Q4	4.816	88.22	59.22	1.720e-2	3.347e-4	cx4_3: 7.193e-3

Your access providers

Provider	Max shots	Max circuits	Usage
ibm-q/open/main	8192	900	View jobs

Figura 2.3: Calibración y características de ejecución del computador cuántico de Santiago

Se puede observar en el apartado de calibración, que esta fue realizada hace, tan solo, doce horas. Como se había comentado, no es extraño que estas computadoras sean calibradas varias veces a la semana, puesto que un mal funcionamiento puede ocasionar que todos los procesos fallen o sean incorrectos. Sobre la información de la calibración se muestran todos los datos divididos por cada qubit, de forma que se pueda intentar predecir el funcionamiento de estos si

se van a realizar cálculos que deben ser altamente precisos.

Para cada uno de los qubits se muestran las frecuencias a la que están trabajando, una serie de tiempos de ejecución y errores de medidas. Además, en la parte de abajo se puede observar los datos del proveedor del sistema, dónde sitúa el máximo de ejecuciones para un proceso en 8192 y el máximo de circuitos ejecutables en 900. Estos datos son importantes para el programador o diseñador, ya que es recomendable realizar el mayor número de ejecuciones posibles para evitar resultados falsos con mayor probabilidad, y no se deben sobrepasar la cantidad máximas de circuitos por trabajo ya que no se ejecutarían y los resultados se obtendrían falsamente.

2.1.4. Diseño de circuitos cuánticos

Conociendo las capacidades que aportan los computadores cuánticos, las cualidades de los qubits y las características de los sistemas cuánticos actuales, se puede pasar a observar los circuitos cuánticos. Estos circuitos son, en efecto, los procesos o trabajos que se encolan en cada uno de los sistemas cuánticos de los que estaba compuesta la nube de sistemas de IBM.

Si se estudian en forma de "caja negra", los circuitos no son más que líneas de programación en un lenguaje especial llamado "QASM" *(Quantum Assembly Language)*. Una vez se comprende el lenguaje, como en cualquier otro lenguaje de programación, se puede empezar a trabajar con él e ir creando, desde circuitos sencillos hasta circuitos muy complejos.

A la hora de conectar esos circuitos con el simulador o la computadora cuántica, solo es necesario tener el circuito construido de forma correcta y utilizar ciertas librerías (de las que se hablará en secciones posteriores). Es necesaria la selección del lugar dónde se ejecutará y la posterior interpretación de los resultados devueltos.

Pero en este apartado del estado del arte no se va a terminar la explicación con unas simples cajas negras que hacen lo que se programa, sino que, se va a pasar a observar algún ejemplo para obtener una imagen visual de cómo se ve un circuito cuántico y, se va a explicar, brevemente, qué significan los principales elementos de un circuito cuántico, y qué función desempeñan en el conjunto de un circuito.

Para ilustrar este ejemplo, se ha seleccionado un circuito cuántico de tres qubits, que se puede apreciar en la imagen.

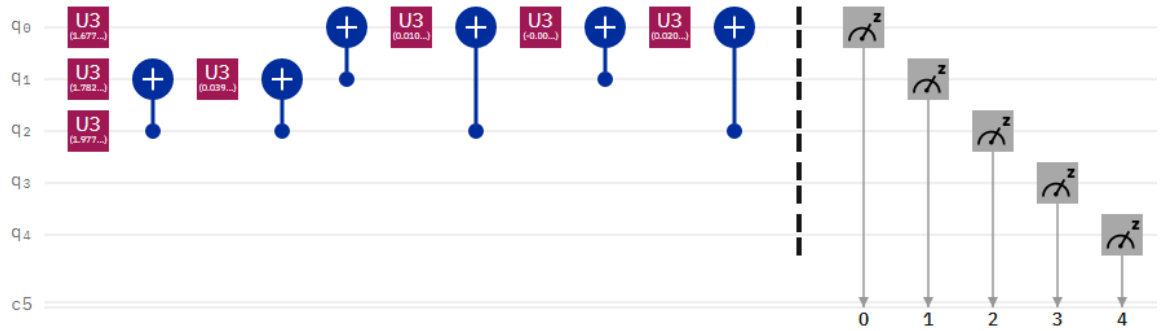

Figura 2.4: Circuito de tres qubits de ejemplo

Hay que destacar que las representaciones del circuito cuántico son las predeterminadas en la página de resultados de IBM, al igual que son las mismas que las del constructor de circuitos, dónde se pueden crear circuitos arrastrando los bloques a los qubits intuitivamente.

Primero hay que observar que el circuito está ordenado cronológicamente, de manera que las

primeras acciones que se realizan en los qubits son las situadas a la izquierda, mientras que las últimas acciones están situadas a la derecha.

El circuito está divido por qubits, que en este caso tiene cinco divisiones al tener esa misma cantidad de qubits el computador cuántico dónde se ejecutó el circuito. Una vez se sabe eso, el siguiente punto importante son las rotaciones de los qubits, que se pueden apreciar con forma cuadrada en cada uno de los qubits. Las rotaciones tienen un posible valor de 0 a 2π, pudiendo utilizarse cualquier rango de valores que mantenga esta distancia. El valor "U3" es el nombre del elemento, mientras que el valor situado debajo es la rotación a la que será sometido el qubit.

Los círculos con cruces blancas en su interior identifican la interacción de un qubit con otro, siendo el qubit que afecta el que tiene el punto en su línea y afectando al qubit que tiene la cruz en la suya. Después de todas las rotaciones y operaciones se aprecia una línea vertical discontinua, cuyo significado es el final de las operaciones entre qubits y la detención de los procesos anteriores para proceder a la medida de los estados. Por último, se puede observar una serie de cuadrados grises que significan la realización de la medida cuántica de cada uno de los qubits, haciéndolos pasar a tratarlos como bits clásicos, dónde puede resultar en un estado o en el otro, cada qubit necesita de su medidor.

Una vez se introduce el circuito en el simulador o en el computador cuántico, este se ejecuta el número definido de veces, produciendo de esta forma que el resultado de la simulación aparezca como una serie discreta de probabilidades, formando un histograma, habiendo sacado los datos estadísticos del cúmulo de ejecuciones realizadas. Se puede observar uno de estos resultados a continuación, del que posteriormente el diseñador debe sacar conclusiones y un resultado concreto.

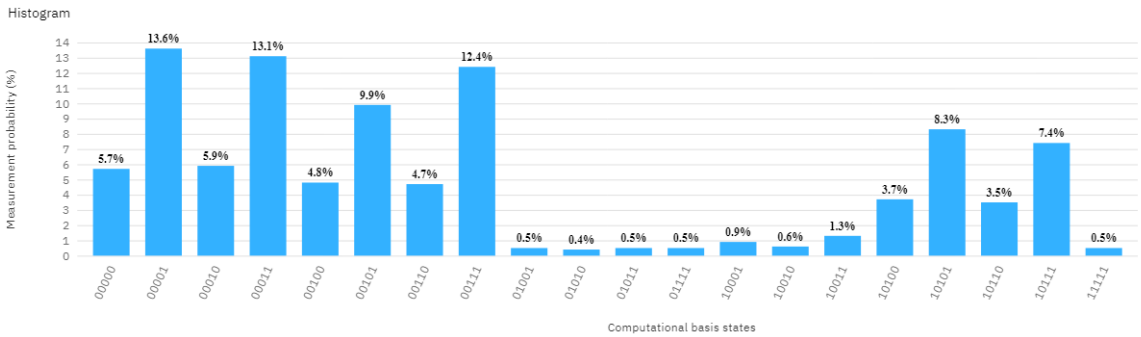

Figura 2.5: Histograma del resultado de la ejecución del circuito cuántico

Dependiendo del circuito ejecutado, del ruido y de los valores de las rotaciones seleccionado, el resultado será de una forma distinta, variando ligeramente incluso cuando se realiza exactamente la misma ejecución, al ser una tecnología probabilística.

2.2. Redes neuronales

La inteligencia artificial ha sido, desde hace más de medio siglo, un ámbito de la computación que ha llamado la atención a curiosos e investigadores. La idea de lograr que un proceso lógico se realice de forma automática, es una de las principales razones por la que esta tecnología es tan impresionante.

Si se revisan los inicios de la inteligencia artificial, encontramos hitos que han ido marcando nuevos avances, tales como:

- Publicación, por Alan Turing, de "*Computing Machinery and Intelligence*". (1950)

- Desarrollo de ELIZA, por el MIT, el primer programa capaz de procesar lenguaje natural y conversar a través de frases programadas. (1966)
- Hans Berliner desarrolla BKG 9.8, una máquina que vence a Luigi Villa al backgammon. (1979)
- El programa "Deep Blue", desarrollado por IBM, gana a Garri Kaspárov en una partida de ajedrez. (1997)
- La universidad de Stanford desarrolla un coche autónomo que gana una competición de vehículos robot recorriendo 212 kilómetros por el desierto. (2005)
- El programa .ªlphaGo", desarrollado por Google DeepMind, vence al campeón mundial de *go*. (2016)
- El programa "Libratus", desarrollado por la Universidad de Carnegie Mellon (EE. UU.) vence en un torneo de *Texas hold'em*. (2017)

Se ha detenido la enumeración de los hitos de la inteligencia artificial en 2017 debido a que, en estos últimos años se han producido tal cantidad de avances que la lista sería inmensa. Tal es el nivel al que está la inteligencia artificial actual que, por ejemplo, algunas de las vacunas creadas para el COVID-19 han sido creadas mediante ella. Pero, ¿qué es exactamente la inteligencia artificial y en qué punto se encuentra en la actualidad?

Se podría definir inteligencia artificial como la capacidad de un programa o máquina de realizar acciones de forma autónoma, sin necesidad de ninguna ayuda. Por ejemplo, un programa que reciba preguntas y devuelva respuestas, o reciba una suma y devuelva el resultado. Esta podría ser utilizada como una definición general, pero bien, en la actualidad, el ámbito de la inteligencia artificial se ha ido abstrayendo de tal forma que, lo único que comparte con los inicios son los principios fundamentales.

La inteligencia artificial ha ido mejorando y transformándose tanto en los últimos años, que hablar del término inteligencia artificial es tan amplio y ambiguo, como sería hablar de matemáticas en general. Por lo que se han ido creando nuevos términos para definir correctamente de lo que se está hablando, los términos van a ser expuestos y explicados con ayuda de una figura ilustrativa.

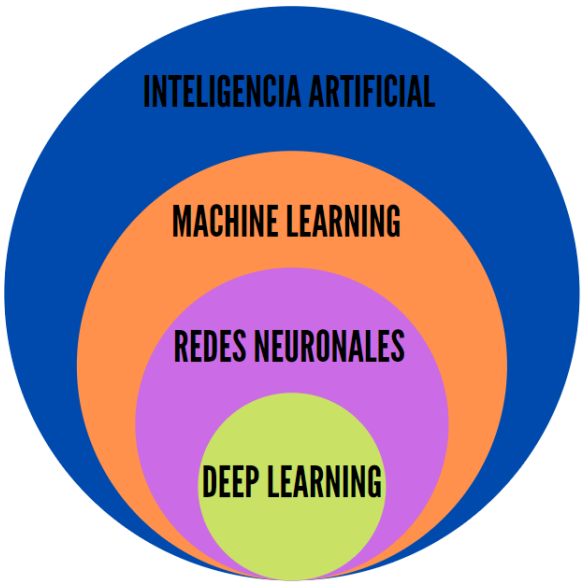

Figura 2.6: Diagrama de la inteligencia artificial.

En la figura anterior se puede observar que, en general, se puede dividir en tres subsecciones, contenidas cada una dentro de la anterior. Ya se ha expuesto lo que significa la inteligencia artifial, pero queda por comentar el significado de las otras tres divisiones:

- Machine learning: Algoritmos que habilitan a computadores a aprender de ejemplos sin estar expresamente programados para ellos. A través del aprendizaje con ejemplos, estos algorítmos son capaces de resolver tareas, sin necesariamente saber cómo lo está haciendo. De forma que se puede encontrar una gran variedad de algoritmos de machine learning que son capaces de resolver tareas que todavía no se saben resolver de forma manual.

- Redes neuronales: Algoritmos inspirados en el cerebro humano con capacidad de aprendizaje de ejemplos. Después de observar los beneficios aportados por el machine learning en el ámbito de la inteligencia artificial, se pensó que una de las formas en las que se podría mejorar el aprendizaje es aplicar los conocimientos que se poseen sobre el cerebro humano, que, hasta la actualidad, no ha sido superado por la inteligencia artificial, dada su enorme complejidad. Se hablará más de este punto después de la enumeración.

- Deep Learning: Los algoritmos dentro del conjunto del machine learning que son capaces de autoajustarse, e incluso, de generar sus propios valores iniciales a través de la unión de varios algoritmos unidos. Uno de los algoritmos de deep learning más populares es el aprendizaje reforzado, donde el programa aprende con cada iteración por sí solo, con una puntuación de la ejecución anterior. Otro de los más populares son los GAN, redes generativas antagónicas, dónde hay dos algoritmos, uno que genera muestras y evalúa el resultado, y otro que es el que realiza el proceso deseado. De esta forma, se puede observar que se tiende a un aprendizaje no supervisado para tareas complejas, ya que pueden ser tareas que ni siquiera se hayan solucionado nunca, como problemas matemáticos.

En este TFM, la inteligencia artificial que se pretende aplicar son las redes neuronales, por lo que se va a exponer estas con una mayor profundidad. Una red neuronal es un conjunto de capas, formadas por neuronas, que son capaces de "aprender" a realizar ciertas tareas. Cada una de las neuronas que forman cada una de las capas son, en realidad, un coeficiente por el que se multiplica el valor que se recibe, para después transmitirlo de salida a la siguiente capa.

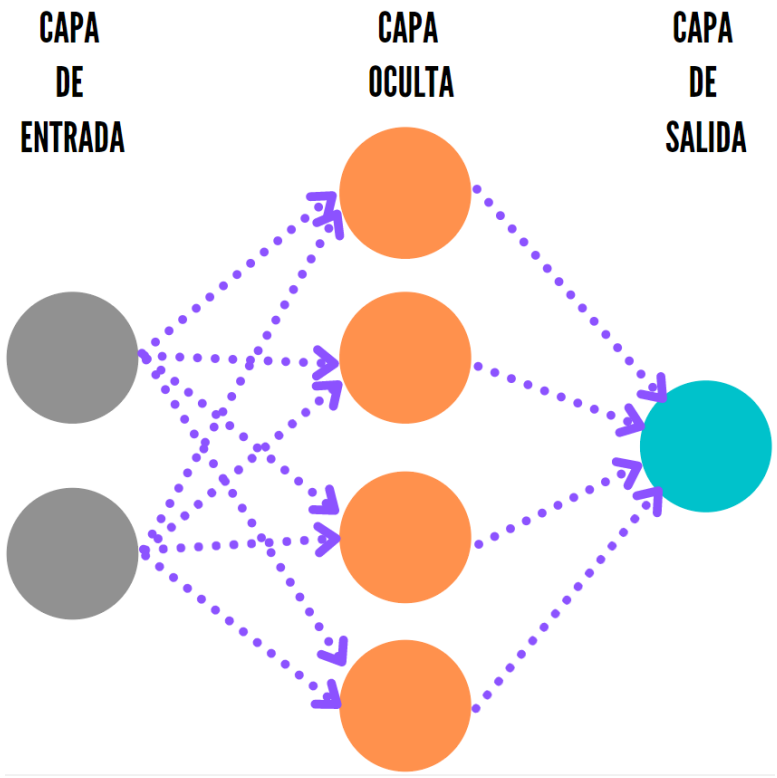

Figura 2.7: Ejemplo del diagrama de una red neuronal.

De esta forma, al unir varias capas y una gran cantidad de neuronas, se alcanzan programas con una complejidad muy alta, de forma automática (previamente diseñados). La decisión sobre la cantidad de capas establecidas o sobre la cantidad de neuronas es trabajo del diseñador, siendo lo más usual tener una idea inicial, que posteriormente será ajustada a base de pruebas realizadas, observando los resultados. Para observar de forma clara esta estructura de las redes neuronales se ha adjuntado una figura ilustrativa.

Inicialmente, el proceso de aprendizaje comienza con la transformación de los datos, para seleccionar las características más relevantes de los mismos para aprender. Hay ciertas ocasiones en las que existen características en los datos que generan ruido en el sistema, empeorándolo en vez de mejorarlo, por ejemplo, si realizamos una red neuronal para calcular el precio de una vivienda, y en los datos tenemos el año en el que nacieron los últimos inquilinos, puede que sea un dato que realmente no tenga ninguna correlación con el precio, y no aporte nada.

Una vez se tienen unos datos para aprender sobre ellos, las redes neuronales se inicializan con unos valores elegidos, que pueden ser de todo a cero, un valor fijo establecido o aleatorios. Este es el punto de partida para el aprendizaje, cuyo valor se mide en un nuevo término, el coste. El coste de una iteración de la red neuronal es el error que comete frente a los resultados que posee como referencia. Esta función de coste es establecida por el usuario, y se dispone de una gran cantidad de costes predefinidos disponibles para su elección, puede ser distancia cuadrática, divergencias o lo que considere necesario.

En el momento en el que se comprende el coste, se debe observar cómo se genera el mismo. Al establecer las capas, el diseñador, establece también qué valores pueden tomar las neuronas que están en la misma, teniendo, de nuevo, una serie de rangos disponibles para su elección, como, por ejemplo, valores de la tangente de números aleatorios o el seno de números aleatorios. Con estas directrices la red neuronal genera los valores en cada iteración y va aprendiendo, intentando reducir el coste.

Cuando se tiene una red neuronal enseñada, lo que se debe hacer es comprobarla con unos

datos de validación, a ser posible, que nunca hayan sido vistos por la red neuronal, para que esta no aprenda a memorizar, sino a comprender la tarea. Cada una de las iteraciones que realiza una red neuronal para aprender recibe el nombre de épocas. Cada época es una ejecución con valores distintos a la anterior, generando un nuevo coste.

Al comprender el proceso de ejecución de las redes neuronales, parece sencillo el entrenamiento, simplemente dejar que la red vaya aprendiendo hasta que el coste sea cero o el menor posible. Pues este pensamiento suele, en la mayoría de los casos, no ser correcto. Puede ocurrir una variedad de situaciones que eviten seguir el proceso intuitivo, por lo que se va a exponer algunos de los casos más habituales que se producen en las redes neuronales y que se deben evitar para un correcto funcionamiento de la red.

Problemas	**Explicación del problema**
Pocos datos de entrenamiento	En ocasiones, el problema que ocurre es que la red no tiene suficientes datos de entrenamiento como para ser capaz de aprender sobre los mismos. Esto se produce cuando los datos iniciales son demasiado escasos para el problema que se pretende realizar. Por ejemplo, es más fácil aprender a sumar, que a calcular el precio de una casa.
Datos *outlayer*	Este tipo de datos poseen valores demasiado alejados del conjunto de datos que se posee, por lo que en vez de aportar un beneficio al sistema, por contra, aportan perjuicio al mismo. Estos datos se encuentran dentro de la base de datos seleccionada como datos iniciales, y es recomendable eliminarlos antes de entrenar a la red neuronal si este tipo de datos resulta ser muy extraño en los casos reales. Por ejemplo, para aclarar este punto, en el mismo algoritmo que aprendiese sobre el precio de la vivienda, si existe una casa de tres millones de euros, y la siguiente más cara es de 200 000€, es recomendable que la casa más cara no entre en los datos de entrenamiento, ya que no se va a aprender nada con ella, sino que va a causar un coste excesivo que corromperá la red.
Sobreentrenamiento	Uno de los problemas más comunes suele ser el sobreentrenamiento de la red neuronal. Intentar que la red alcance un coste cero es un grave error, puesto que, si se diese este caso, la red aprendería a resolver perfectamente los casos de entrenamiento, pero probablemente no sería capaz de resolver casos que nunca se le hubiesen presentado. De hecho, ni siquiera es necesario que la red alcance el coste cero para que sea incorrecto, se puede observar en el funcionamiento de las mismas que, después de un cierto número de épocas, la red empieza a memorizar los casos de entrenamiento, perjudicando el funcionamiento general.

Continúa en la siguiente página.

Cuadro 2.1 – *Continuado de la página anterior.*

Problemas	**Explicación del problema**
Mala transformación de los datos	Los datos iniciales, además de tener que ser recolectados y filtrados para eliminar algunos de ellos, necesitan de una transformación. Para la correcta transformación de los datos es necesario observar si estos tienen correlación entre ellos, a veces, con tan solo dos características de los datos se consiguen mejores resultados que con cinco, es cuestión de seleccionar los adecuados. Para esta selección lo que se suele hacer es un proceso lógico de análisis de las características de los mismos, y posteriormente, un ajuste en función de los resultados que se van obteniendo en las ejecuciones.
Falta de entrenamiento	Al igual que se puede sobreentrenar a la red neuronal, puede darse el caso de que para el entrenamiento se hayan utilizado demasiadas pocas épocas, o que el entrenamiento la red el entrenamiento se haya ejecutado una sola vez, pudiendo caer en un mínimo local en vez de encontrar el mínimo absoluto del coste.
Desequilibrio entre las características de los datos	A parte de esta serie de problemas, a la hora de recolectar los datos y seleccionar los que servirán de entrenamiento y los que servirán de validación, se encuentra otro problema. Para que la red neuronal aprenda de forma correcta, se necesita que esta serie de datos tenga un equilibrio entre sus características diferenciadoras. Por ejemplo, en una red que intente diferenciar imágenes de gatos con otras de perros, si en la base de datos de entrenamiento hay cien gatos y solo un perro, es altamente probable que la red neuronal no sepa diferenciarlos de forma adecuada. El caso del ejemplo, es algo sencillo, pero incluso ese caso se podría complicar, dividiendo los perros y gatos por tamaño, raza, color, etc.

Cuadro 2.1: Problemas comunes en redes neuronales.

Con todos estos problemas posibles en las redes neuronales, siendo estos los más comunes, se observa que no resulta algo trivial, sino un tipo de algoritmos que suelen requerir de un profundo razonamiento y un trabajo de transformación de datos y análisis de resultados exhaustivo.

Aunque las redes neuronales aportan unas capacidades impresionantes, dada su amplia complejidad y las posibilidades que aportan, en la actualidad, el deep learning es la parte de la inteligencia artificial que más está expandiéndose, ya que, al aportar la posibilidad de que las redes se auto entrenen o sean ellas mismas las juezas de otras, permite realizar tareas muy complejas de forma autónoma. Día tras día, también con el avance en la computación, se habilita la creación de redes más complejas que mejoran a sus predecesoras, o resuelven problemas que estas no podían resolver.

En la actualidad, como dato curioso, las personas, y más concretamente, los usuarios de programas informáticos o de internet, están rodeados de inteligencia artificial por todas

partes, desde el algoritmo que decide qué correos son spam en base a sus decisiones, como las recomendaciones de compras que se producen en los anuncios de las páginas web.

2.3. Riesgos financieros

En esta sección se va a pasar a explicar el estado de arte de los riesgos financieros que calculan las entidades bancarias, así como, las definiciones básicas necesarias para lograr comprender el objetivo global del TFM.

Los conceptos que se explicarán también ayudarán a aquellos lectores que no conozcan el ámbito financiero a tener una idea de las definiciones básicas para comenzar a adentrarse en el mundo matemático de los riesgos financieros. No se pretende que estas explicaciones sean, ni exhaustivas, ni demasiado complejas, simplemente definir las bases de los riesgos financieros.

El riesgo financiero podría definirse como la posibilidad de que ocurran eventos adversos que causen consecuencias negativas o positivas, afectando en este caso a la economía de una entidad bancaria, un inversor o empresa. Estos riesgos tratan de ser evitados en la medida de lo posible, puesto que sus efectos pueden ser devastadores. Cuanto mayor sea el efecto que causen y la probabilidad de que sucedan, más se intentará evitarlos, de forma que si un riesgo financiero no tiene graves consecuencias y la probabilidad de que ocurra no es alta, puede ser que no se intente evitar, y se enfoque la protección ante otros riesgos distintos.

Para las entidades bancarias, unos de los riesgos financieros que deben tener en cuenta es el VaR estimado de los activos bancarios. Con ese valor, pueden manejar el porcentaje que variarán esos activos al día siguiente, lo que permite tener una cierta ventaja de la que sacar partido en el momento de tomar decisiones. Si esta variación es cercana al VaR estimado, resulta muy beneficioso, pero si por el contrario el valor de la variación es lejano a él, resulta ser un problema.

En muchas situaciones, resulta que ciertos activos se pueden tratar de una forma correlacionada, debido a su influencia ante otras, por lo que facilita que el cálculo global de ellas sea más cercano a la variación real, y, por tanto, permite una mayor precisión. En el caso de este TFM, lo que se va a calcular en el programa final es el VaR con una confianza del 95 %, del conjunto de activos correlacionados seleccionados.

2.3.1. Métodos de cálculo

En esta subsección se hablará de los métodos actuales más utilizados para calcular el VaR (valor de riesgo), es decir, la variación o fluctuación esperada de un activo. Este VaR es predicho con suficiente margen como para que la probabilidad de perder dinero sea tan solo de un 5 %.

Para aclarar esta explicación se ha decidido incluir una imagen, de forma que al realizar una explicación sobre una figura visual, esta se entienda rápidamente.

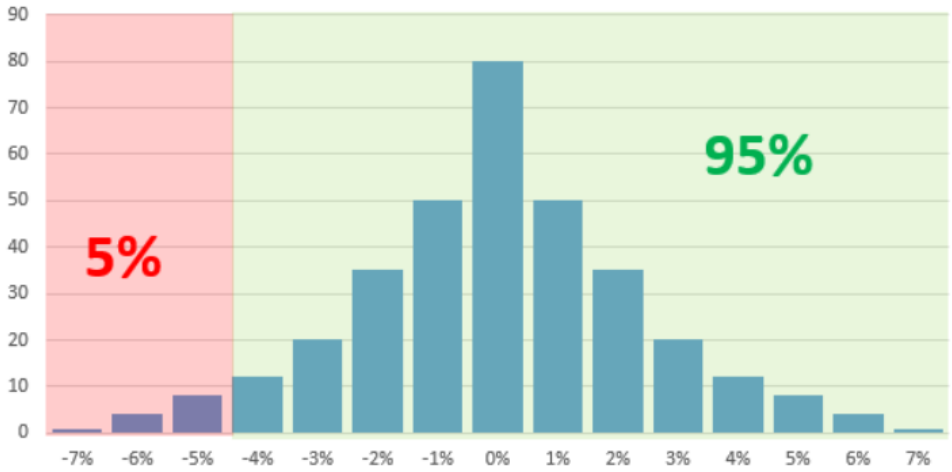

Figura 2.8: Ejemplo de un histograma para calcular el VaR[16][17].

Se puede apreciar en la figura una zona coloreada en rojo, dónde se sitúa el valor de "5 %" y otra zona en color verde dónde se sitúa el valor "95 %". Si se observa el histograma, se puede observar la zona roja está colocada desde el 0 % de la distribución hasta el 5 %, quedando el resto de la probabilidad en la zona verde.

Estas zonas tienen un significado económico, si se coloca el valor de riesgo justo en el punto que ambas zonas se encuentran, resulta que, si las pérdidas de la acción es uno de los valores de la zona roja (suceso que ocurrirá con un 5 % de probabilidad) perderá dinero la entidad bancaria, mientras que, si el valor real que sucede es uno de los de la zona verde, la entidad ganará dinero.

De esta forma, colocando el riesgo al 5 %, hay pocas probabilidades de que la entidad bancaria pierda dinero, y además, aunque se perdiese dinero, este sería poco, mientras que, al contrario, las ganancias se maximizan, puesto que si se predice un valor del -4 %, como en el caso de la figura, y resulta ser un valor final del 7 %, las ganancias se maximizan (Aunque este suceso es también muy poco probable).

El cálculo de este valor de riesgo, para que se maximicen las ganancias y se minimice el riesgo se calcula de distintas formas, siendo las más utilizadas estas:

- VaR histórico:

  El método de estimación del VaR histórico es el único que calcula un VaR pasado, es decir, este método no intenta estimar el valor del día siguiente, sino que obtiene un valor antiguo a través de las pérdidas diarias de los activos. Para poder calcular el VaR histórico se coge las cantidad deseada de pérdidas de los activos y se calcula el percentil 5 %, en este método ese es el VaR resultante.

  Este método tiende a reaccionar a los cambios lentamente, debido a que es necesario que cambien una amplia cantidad de valores para que varie el resultado significativamente. Pero como beneficio tiene que es un método de bajo coste computacional, por lo que se puede realizar sobre una cantidad amplia de pérdidas de activos. Además en este método no se asume ninguna aproximación sobre el tipo de distribución de datos.

  El método histórico es muy importante en el diseño final implementado, ya que se utiliza para calcular los valores de salida iniciales de las agrupaciones de los valores de pérdidas de los activos.

- Método paramétrico:

  Este método permite realizar el cálculo de la variación del riesgo de activos. Se trata de un método aproximado, ya que intenta estimar una variación sin saber exactamente el valor de la misma.

  Para esta aproximación se utilizan los datos de rentabilidad estimados y se asume una distribución normal de la probabilidad. De esta forma, se pueden realizar los cálculos necesarios para obtener el valor de riesgo, de una forma relativamente sencilla (si se poseen todos estos datos y valores).

  Al tener todos los datos de rentabilidad esperados y el riesgo histórico (medido a través de la desviación típica) se aplica la siguiente ecuación para calcular el valor de riesgo:

  $$VaR = |R - z\delta|V$$

  Donde R, representa la rentabilidad esperada, z, es el valor correspondiente para un nivel de significancia dado (se debe fijar), δ, es la desviación típica de la rentabilidad esperada y por último, V, es el valor de la inversión realizada.

  El método paramétrico suele ser el método más fácil de calcular si se tienen estos datos, pero a la vez es menos preciso que el método de Montecarlo, lo que establece una relación "precisión-realizabilidad", dónde se debe hallar el punto medio para satisfacer ambas necesidades. Además de que no se tienen en cuenta los posibles sucesos que podría ocurrir (unicornios y cisnes negros).

  Por realizar estos cálculos suponiendo la distribución normal de probabilidad, este método también es conocido como método varianza-covarianza o método analítico.

- Método de Montecarlo:

  Este método es el más utilizado en el estado del arte actual, ya que, al tener una mejor precisión que el método paramétrico, las entidades bancarias lo prefieren para tener menor riesgo, y, por ende, perder menor capital. Para el cálculo del mismo, se requieren de una serie de valores iniciales con los que se realizarán distintas fórmulas matemáticas, para después, generar una serie de valores "aleatorios" con los que se calculará el valor de riesgo.

  Al haber generado los valores aleatorios, se puede seleccionar el valor de riesgo como el percentil 5 % de la distribución generada. Un problema de este método es, que si los valores iniciales utilizados para generar la distribución aleatoria no son correctos, el método cometerá un importante error en el resultado final.

  Se podría utilizar para ilustrar este ejemplo la misma figura que en el método paramétrico, pero resulta que este ejemplo es mucho más visual, ya que los pasos necesarios para obtener el resultado son, realizar los cálculos iniciales, y una vez generada la distribución aleatoria, ordenar los valores y obtener el resultado en función del grado de confianza que se desee, cuyo valor suele ser del 95 %, por lo que quedaría el 95 % de la distribución de probabilidad en la zona de ganancias.

  Hay que destacar que de forma general este método obtiene un valor de riesgo

mayor que la distribución paramétrica, por lo que se acerca más al valor de riesgo histórico que se puede calcular una vez se conoce toda la distribución real. Además este métido si tiene en cuenta los posibles sucesos improvistos que pueden afetar al VaR (unicornios y cisnes negros).

2.4. Librería Qiskit (Python)

En esta sección se va a hablar de la librería escogida para la creación y el manejo de los circuitos cuánticos, la librería Qiskit de Python. Primero se debe comentar, ¿Por qué esta librería se encuentra en Python? En la actualidad Python es un lenguaje de programación que se encuentra en alza, llegando poco a poco a un gran abanico de tecnologías, debido a su facilidad de utilización al ser un lenguaje interpretado y a su sencillez. Estas cualidades hacen que cada día más gente utilice el lenguaje, permitiendo formar una comunidad que va mejorando día a día, y cuyas habilidades son capaces de facilitar las labores a los demás usuarios a través de la creación de librerías de código abierto.

Al momento de ir a sumergirse en el mundo cuántico, Qiskit resulta ser una librería muy útil, puesto que facilita la construcción, la transformación de los circuitos al lenguaje "QASM" y la ejecución de los mismos. La librería Qiskit es considerada la librería estándar en el ámbito de la computación cuántica.

Esta librería está compuesta por tantas funciones, y aporta tantas posibilidades a la hora de crear y administrar circuitos cuánticos, que está dividida en cuatro apartados, denominados como los cuatro elementos de la naturaleza: Terra, Aer, Ignis y Aqua. En estas cuatro divisiones se encuentran distintas características de la librería, habilitando una enorme variedad de acciones y actualizándose continuamente para seguir implementando más posibilidades cada día. Por si se desea introducirse en el mundo de la computación cuántica y comenzar a utilizar la librería, en las referencias se puede encontrar directamente la dirección web para comenzar.

En general, como a la hora de utilizar cualquier otra librería, la mejor forma de familiarizarse con ella es, comenzar a probar y realizar programas simples, por lo que no se va a profundizar en todas las funciones y las posibilidades que otorga esta librería, pero se quiere destacar alguna de ellas por su elevada importancia:

Funciones	**Propiedades y acciones que realizan**
Provider.backends()	Esta función permite al usuario obtener todos los computadores cuánticos disponibles para realizar al ejecución de los procesos. Dependiendo de si el usuario selecciona directamente con una cadena de texto, o si utiliza las propiedades de la función para que se facilite la elección, será elegido uno u otro. Esta función también depende del "token" del usuario para conocer a qué sistemas tiene permitido acceder, y diversas propiedades para seleccionar por número de qubits o por menor número de trabajos encolados.

Continúa en la siguiente página

Cuadro 2.2 – *Continuado de la página anterior*

Funciones	**Propiedades y acciones que realizan**
Least_busy()	Esta función sirve para realizar una elección automática del computador cuántico que menos trabajos encolados tiene de una serie de sistemas que se le introducen a la función. Cómo se puede suponer esta función sirve para ampliar las capacidades de la función anterior, y suele contenerla para realizar la elección óptima.
NormalDistribution()	Esta función realiza el papel de generador de la distribución normal, que pasará a ser utilizada para el cálculo del resultado en las ejecuciones. Es capaz de generar una distribución normal, estableciendo su límite superior en un valor fijado, al igual que su límite inferior. Con ese rango de valores, crea una distribución normal con los valores "sigma" y "mu", y posteriormente se dividen los valores equitativamente según el número de qubits seleccionados, por ejemplo, si el valor de qubits es tres, la normalización se dividirá en ocho valores distintos de la distribución de densidad de probabilidad.
Execute()	Esta función es la que habilita la ejecución de los circuitos que se han diseñado. Es capaz de recibir un circuito, un sistema donde se desee ejecutar el mismo y devolver después de su ejecución el resultado correspondiente. Si se selecciona ejecutar el circuito en un sistema real, la función no terminará hasta que el proceso atraviese la cola de programas y se ejecute correctamente, poniendo en espera el programa en caso de continuar después.
Qc.qasm()	Esta función desempeña el papel de traductor desde un circuito construido en lenguaje Qiskit al lenguaje QASM. Esta transformación permite crear desde circuitos predeterminados, circuitos personalizados eligiendo las rotaciones de los qubits.
QuantumCircuit.from_qasm_str()	Esta función realiza una transformación inversa a la anterior, pasa desde un circuito en lenguaje QASM a lenguaje Qiskit, para poder ejecutarlos posteriormente. En este lenguaje se permiten las ejecuciones y el circuito se construye para manejarlo de forma "compacta", mientras que tenerlo en QASM permite personalizarlo más fácilmente.

Continúa en la siguiente página

Cuadro 2.2 – *Continuado de la página anterior*

Funciones	**Propiedades y acciones que realizan**
Qc_nuevo.draw()	Esta función permite dibujar el circuito cuántico realizado, pudiendo apreciarse sus puertas, acciones o rotaciones de los qubits. Hay que destacar que, dependiendo de la versión de la librería, el dibujo del circuito es de una forma u otra, pero con las mismas características y resultados.

Cuadro 2.2: Funciones destacables de la librería Qiskit.

Esta serie de funciones son algunas de las más utilizadas en la realización del TFM. Se ha decidido exponer las funciones sin profundizar demasiado en tipos de datos y resultados. Se puede aprender más sobre las mismas en la propia documentación oficial de Qiskit, dónde se pueden encontrar ejemplos y guías para aprender a utilizarlas.

3
Diseño

En este capítulo se van a tratar las decisiones y los distintos métodos planteados para abordar el desarrollo del TFM. Una vez se tengan claras las ideas iniciales, se podrá pasar a observar el desarrollo de todas las ideas, con sus comportamientos esperados, los resultados de las mismas y los problemas encontrados, pero esto se verá en el siguiente capítulo para diferenciar la parte de desarrollo de las necesidades previas a él.

Antes de comenzar a definir la serie de tareas a realizar para llegar al objetivo final, se deben tener claras ciertas decisiones previas, tales como el planteamiento inicial con el que se va a abordar el diseño final, para poder realizar las tareas que nos vayan llevando al mismo. Si se logra realizar un diseño de tareas con dificultad escalonada y que a su vez se van ampliando levemente, se logra obtener el programa en la versión final sin necesidad de empezarlo desde cero.

Por estas razones, es necesario detener el proceso de realización del TFM una vez se tienen los conocimientos iniciales necesarios, y pensar de forma detenida los pasos que se van a seguir a continuación. Si se estableciesen tareas muy complejas o demasiado extensas, no se cumpliría con el objetivo de horas propuesto, ni con la fecha de realización estimada.

3.1. Planteamiento inicial

Al inicio del TFM, como en cualquier otro proyecto de investigación, los primeros pasos son informarse del estado del arte y conocer las tecnologías y los métodos que se están utilizando actualmente para llevar a cabo las tareas a las que se pretende aportar un beneficio.

Por lo que los primeros pasos, tal y como se plasma en el apartado de fases de realización de la introducción, fueron las lecturas sobre la computación cuántica. Una vez se poseía una base sólida sobre los fundamentos de la computación cuántica, el siguiente paso era aprender sobre la realización de los métodos de cálculo del valor de riesgo, y cuál de ellos era el más utilizado actualmente.

El siguiente paso que se ha seguido, ha sido el de informarse sobre las posibilidades que habilita la librería Qiskit de Python y la lectura de la documentación para aprender sobre la misma. También se realizaron varios tutoriales previos y algunos ejemplos para comprobar el

funcionamiento y el manejo de los tipos de datos necesarios para las mismas. Se profundizará sobre estos pasos más prácticos en el apartado del desarrollo del TFM.

Una vez leído y comprendido todo esto, se pasó a observar las necesidades para alcanzar el programa final. En primer lugar, se necesitaba un entorno en el que empezar a programar, dónde se pudiesen realizar ejecuciones y que además permitiese ejecuciones en un computador cuántico real para las comprobaciones finales,

Lo primero al ir a diseñar los pasos iniciales, es tener en mente el punto final que se pretende alcanzar, por lo que antes de avanzar hay que realizar una revisión del objetivo global del TFM. Para poder observar todos los factores necesarios para su desarrollo, la mejor forma es realizar un esquema visual, sobre el que comentar estos pasos.

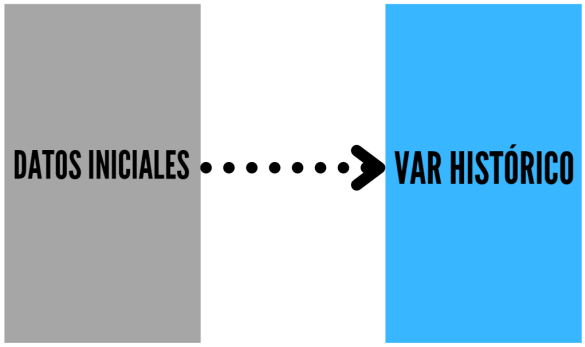

Figura 3.1: Planteamiento de la obtención de los datos iniciales del programa.

Una vez se obtienen los datos iniciales para realizar todo el proceso, se necesita de un procesado de los mismos para obtener el VaR histórico de las distribuciones. Esto es necesario debido a que se necesita un valor "fiable" al que compararse para medir la precisión de los diversos procesos creados.

Este paso, aunque sea necesario para el programa final, no será abordado hasta los últimos pasos, debido a que primero se necesita manejar el entorno cuántico y tener la posibilidad de crear y ejecutar diversos circuitos y programas. Pero en este punto, la idea más probable es la de realizar estos cálculos previos a la ejecución y pasárselos como datos iniciales al programa final, de forma que sean previamente preparados y almacenados en ficheros.

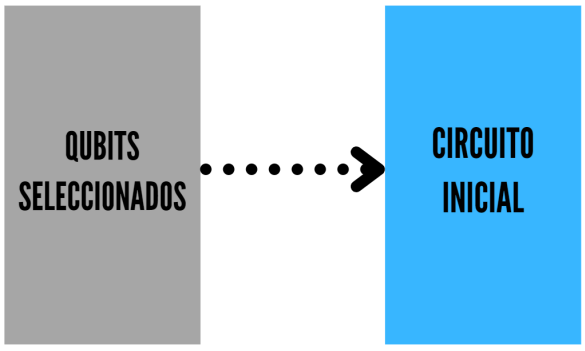

Figura 3.2: Planteamiento de la creación del circuito inicial.

Se pretende que el usuario del programa final sea capaz de cambiar el circuito inicial utilizado dependiendo de los qubits que decida utilizar, de esta forma se espera que el circuito se ajuste a las variables solicitadas por el usuario. Esta decisión posibilita la adaptación del programa a diversas cantidades de qubits y, por tanto, a poder mejorarse o ampliarse sus capacidades si así se desease.

Sin embargo, esto puede producir una complicación, ya que al variar el circuito inicial, el aprendizaje también debe variar con la nueva cantidad de puertas, y, por tanto, rotaciones de los qubits correspondientes. Estas complicaciones deberán tenerse en cuenta a la hora de diseñar el aprendizaje, para que se implemente todo conjuntamente y esta variación no cause ningún percance al programa en general.

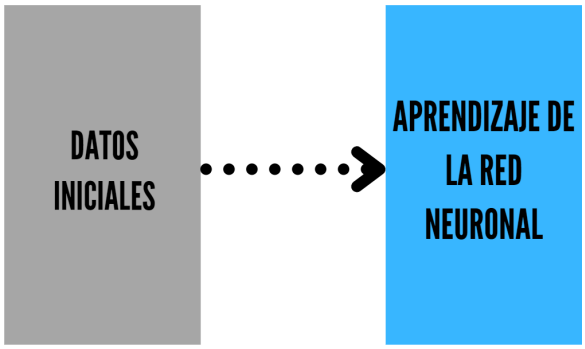

Figura 3.3: Planteamiento del aprendizaje de la red neuronal.

Con los datos iniciales, también se pretende implementar el aprendizaje de la red neuronal. Este aprendizaje dependerá de los datos iniciales de los activos y de la cantidad de qubits empleada en el circuito cuántico. A parte de la administración del dimensionamiento de la red neuronal, el tipo de capas y la cantidad de las mismas, hay que tener en cuenta la mayor complicación, habilitar la ejecución en el computado cuántico.

Esta dificultad reside en que se necesita crear una nueva función de coste para la red neuronal, de forma que el coste con el que aprende la red neuronal no salga de las funciones predeterminadas, y, por contra, se calcule el resultado con el que se calculará el coste en la computadora cuántica. Es decir, se necesitará que la red neuronal aprenda a conseguir las rotaciones de los qubits que satisfacen las necesidades, para que posteriormente, en la función de coste personalizada, se ejecute el circuito cuántico diseñado y poder obtener así el resultado de la ejecución que se deberá comparar con el VaR histórico proporcionado.

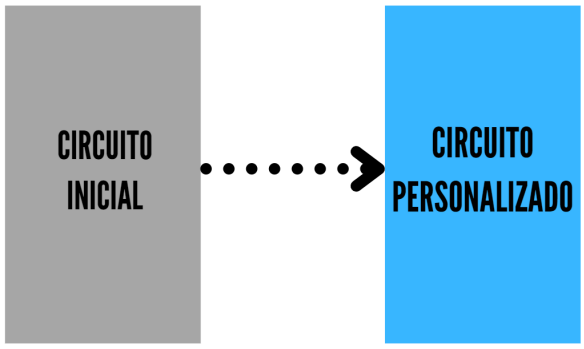

Figura 3.4: Planteamiento de la personalización del circuito inicial.

Es necesaria también la creación de una función que habilite la transformación de los circuitos iniciales generados por el programa a un circuito personalizado, donde se puedan alternar cada una de las rotaciones de los qubits.

Para realizar esto será necesario, de nuevo, tener en cuenta la posibilidad de estos circuitos de variar ante un cambio de qubits, por lo que se necesitará que se posea la capacidad de personalizar circuitos de forma universal. Además de esta necesidad, también es conveniente habilitar la personalización de una sola parte del circuito cuántico, manteniendo los demás

valores invariantes. De esta forma se puede observar el cambio que realiza en los resultados variar una sola rotación de los qubits. Se puede ir probando con distintas cantidades de qubits variadas o el valor que cambian.

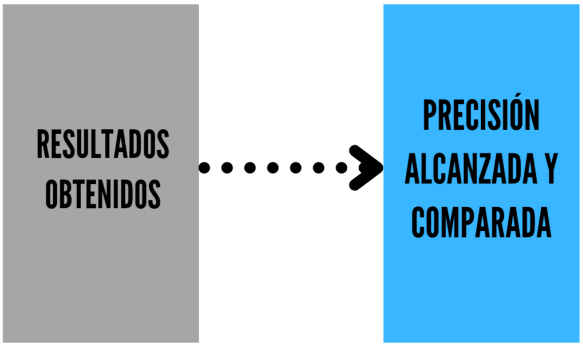

Figura 3.5: Planteamiento de la obtención de la precisión del programa final.

Una vez se obtengan los resultados de la ejecución en la función de coste, será necesaria realizar la comparación con los datos del VaR en los datos iniciales, de forma que se pueda obtener la precisión alcanzada con el aprendizaje y las distintas pruebas realizadas.

Toda esta serie de tareas previstas se irán realizando a través de pequeños pasos, para poder así obtener un programa complejo desde una serie de pasos más sencillos. En el capítulo de desarrollo se relatarán todos los pequeños pasos seguidos en el proceso del proyecto, mientras que el programa final será explicado en otro capítulo, dada su gran complejidad frente a los pasos anteriores.

Al tener una idea prevista de los pasos a realizar en el desarrollo del TFM, lo siguiente que se necesita es saber dónde se va a llevar a cabo ese desarrollo, de forma que se encuentre una plataforma de utilidad que habilite la realización de los mismos.

3.2. Plataforma de desarrollo

En esta sección se va a exponer el motivo de la elección de la plataforma de desarrollo y las capacidades que se habilitan en la misma.

Al haber definido previamente las necesidades, que aparentemente poseerá el diseño del programa final, se puede tener un idea de las capacidades que debería tener la plataforma de desarrollo para habilitar el correcto desarrollo y funcionamiento del mismo.

Esta elección fue más una revisión de las características de una plataforma en concreto, ya que se había trabajado con ella previamente para hacer alguna prueba por parte del tutor y el ponente del TFM. La plataforma, en cuestión, es la plataforma de IBM (*IBM Quantum Experience*).

Esta plataforma de desarrollo, creación y administración de circuitos cuánticos tiene unas amplias capacidades para la computación cuántica. Para observar todas las características necesarias, se ha decidido enumerarlas en una pequeña tabla ilustrativa.

Requisitos	**Causa de esta necesidad**
Capacidad de ejecutar en computadoras cuánticas reales	Para elegir esta plataforma de desarrollo, es necesario que se puedan ejecutar circuitos cuánticos en sistemas cuánticos reales, ya que si se necesitase la conexión con otra proveedora para la ejecución el programa se complicaría sin ningún motivo.
Capacidad de construir circuitos de prueba	En este caso no se trata de una necesidad para el programa final, sino de una necesidad para las tareas iniciales del desarrollo del TFM. Esta capacidad de construir en la propia plataforma facilita en gran medida el aprendizaje sobre el diseño de circuitos cuánticos y los elementos que los forman.
Capacidad de desarrollar programas	Para poder realizar los diversos pasos y programas es necesario poseer un entorno de desarrollo, por lo que era necesaria esta capacidad de la plataforma, que, en concreto, habilitaba la creación de programas en el lenguaje Python bajo el entorno de Jupyter. Esta capacidad previa se verá superada en el programa final y se necesitará estudiar otras opciones, problema que se verá en el capítulo cinco.
Buen rendimiento en velocidad computacional	Para el correcto desarrollo de los pasos es necesaria una alta velocidad computacional. Como se ha visto en apartados anteriores, estos proveedores tienen sus servicios alojados, como los proveedores actuales en la nube, por lo que tener máquinas cercanas y una buena velocidad con ellas es crucial. El proceso de desarrollo se atrasaría en gran medida si cada ejecución de un proceso llevase un largo tiempo.
Poseer un simulador cuántico	A parte de ser capaces de ejecutar los programas en computadores cuánticos reales, es necesario que la plataforma disponga de la posibilidad de ejecutar los circuitos diseñados en un sistema simulado para poder comparar con el real. Esto es básico dado que uno de los factores limitantes a la hora del diseño es el ruido en las máquinas reales.
Capacidad para ver los resultados de forma clara	Es de gran utilidad que la propia plataforma permita al usuario observar de forma clara y precisa los resultados de las ejecuciones. Esta capacidad facilita en gran medida la comprensión de los circuitos y además permite comprobar los cálculos locales, de forma que, si los resultados coinciden, significa que se están calculando de forma correcta.

Continúa en la siguiente página.

Cuadro 3.1 – *Continuado de la página anterior.*

Requisitos	**Causa de esta necesidad**
Capacidad de almacenar los resultados	Una capacidad extra a tener en cuenta es la de almacenar resultados obtenidos. A la hora de realizar pruebas, es conveniente poder observar los resultados anteriores, o, incluso poder observar otros procesos ejecutados para asegurarse del correcto funcionamiento del sistema.

Cuadro 3.1: Capacidades necesarias de la plataforma de desarrollo.

Una vez se revisaron todas estas capacidades en la plataforma de IBM, se optó por comenzar el desarrollo en la misma. Aprovechando de esta forma todo el bagaje del conocimiento adquirido previamente sobre ella.

3.3. Diseño de circuitos de prueba

Para obtener una idea inicial de cómo debían ser los circuitos cuánticos que se iban a personalizar y en qué medida afectaban las rotaciones al resultado de la ejecución, se decidió realizar una serie de pruebas con circuitos cuánticos iniciales. Estas pruebas consistieron en ejecuciones de circuitos predeterminados, circuitos personalizados y circuitos creados íntegramente.

En primer lugar, se comenzó con la ejecución y revisión de los resultados de los circuitos iniciales, estos circuitos son ejemplos por defecto de la librería, y permiten hacerse una idea del aspecto que poseen los circuitos cuánticos y las rotaciones que suelen tener los mismos. Una vez entendido los bloques que suelen poseer, se pasó a observar los resultados y a intentar extraer de ellos el mayor conocimiento posible.

Posteriormente, se pasó a la personalización de dichos diseños. Esto se pudo llevar a cabo replicando los circuitos iniciales y cambiado los valores de las rotaciones de los qubits establecidas por defecto, de forma que cada ejecución era una nueva ejecución, con un resultado diferente al anterior. De este modo se pudo comprender en qué grado cambian los resultados cuando se varían los qubits y las diferencias que tiene variar uno u otro.

Por último, se pasó a realizar circuitos propios. La realización de esta serie de circuitos creados íntegramente se llevó a cabo con los conocimientos que se habían adquirido de los casos anteriores(circuitos iniciales y personalizados). Se pudo llevar a cabo gracias al constructor de la plataforma de IBM, herramienta que permite el diseño de circuitos cuánticos de una forma sencilla e intuitiva.

Esta herramienta habilita la creación de los circuitos arrastrando los bloques deseados a los qubits elegidos, pudiendo realizar un diseño complejo rápidamente y entendiendo la función de cada uno de los bloques utilizados. Además, esta herramienta permite ejecutar los circuitos creados, de manera que se puede crear un circuito, ejecutarlo, y posteriormente, realizar los cambios necesarios en el mismo para ajustar su comportamiento al deseado.

Para realizar estos ajustes correctamente, lo que se debe hacer es ejecutar el circuito cuántico en el simulador y en un sistema cuántico real. Las diferencias que se podrían producir por el

ruido podrían ser tales, que el circuito necesite cambiar para poder imitar el comportamiento del simulador en el real.

3.3.1. Diferencias observadas entre la ejecución en simulador y real

Como apartado final antes de comenzar el desarrollo, se debe realizar un ejemplo del problema del ruido en la ejecución real de los circuitos cuánticos. Ya se ha comentado en varias ocasiones que, al ejecutar un circuito en un computador cuántico, se produce un ruido debido a las diminutas vibraciones de los qubits y a las diferencias de temperatura de los mismos.

Actualmente, evitar estas vibraciones o cambios de temperaturas totalmente es algo irrealizable, por lo que se debe tener en cuenta al diseñar los circuitos cuánticos, e intentar minimizar el efecto del mismo sobre las ejecuciones. Este proceso puede ser muy complejo y es uno de los problemas sobre los que más se está investigando actualmente, puesto que, si se logra realizar ejecuciones en sistemas reales como los haría un simulador, la capacidad de desarrollo aumentaría significativamente.

Para mostrar los efectos reales del ruido sobre un circuito cuántico, y la variación en los resultados, se ha decidido mostrar el circuito y los resultados del mismo, en el simulador y en un sistema cuántico real.

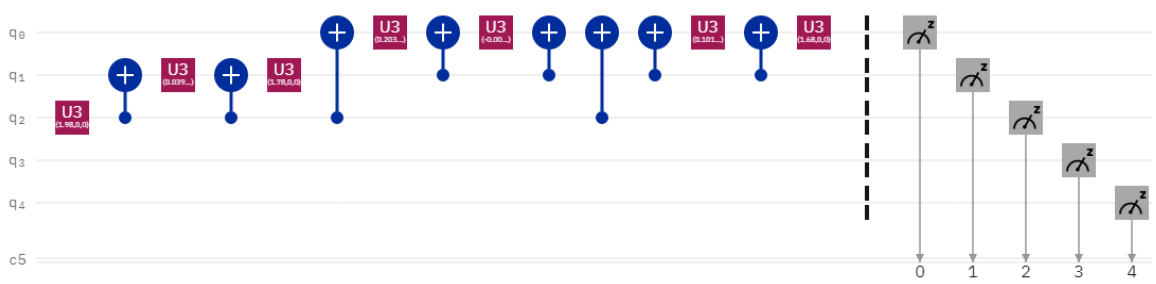

Figura 3.6: Circuito cuántico de ejemplo.

Se puede apreciar que, en este caso, se trata de un circuito que utiliza tres qubits, con diversas operaciones, posteriormente una barrera de fin de ejecución y unos medidores que pasan del estado de superposición los qubits a estados clásicos para sacar resultados.

En el primer caso, se va a ejecutar el circuito cuántico en el simulador de la plataforma de IBM, dónde se observará el histograma de los posibles estados de las ejecuciones. Se ha situado el número de ejecuciones en el máximo admisible de 8192, para poder obtener los resultados más precisos posibles.

En el segundo caso, se ejecutará el circuito en el computador cuántico real de Santiago, con el mismo número de ejecuciones que el simulador. Se van a mostrar ambos resultados seguidos para poder apreciar las diferencias de forma inequívoca.

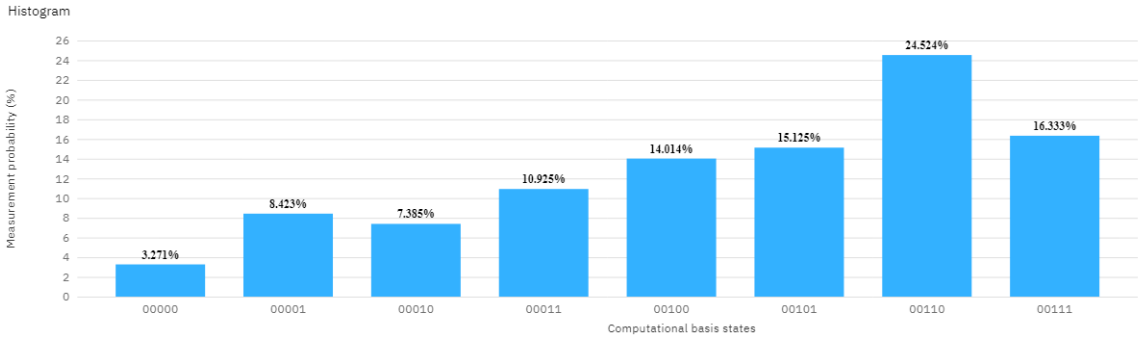

Figura 3.7: Resultado con el simulador del circuito cuántico de ejemplo.

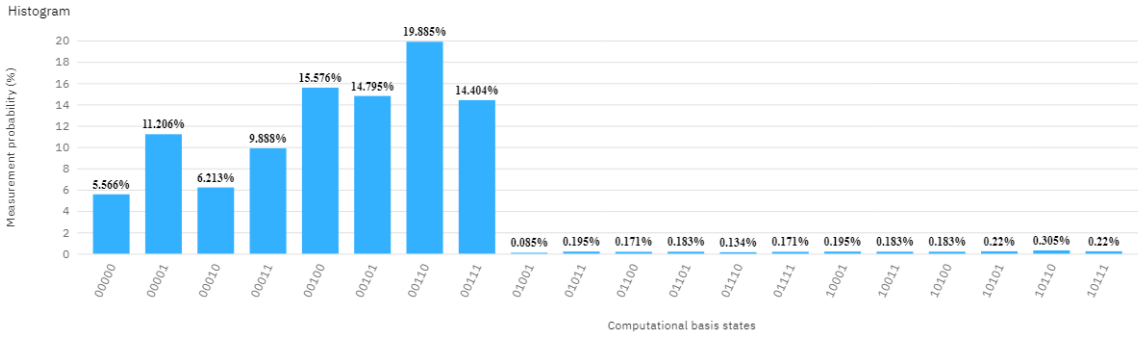

Figura 3.8: Resultado con el sistema real de Santiago del circuito cuántico de ejemplo.

Se puede observar, claramente, las diferencias en los resultados obtenidos entre el simulador y el computador cuántico real. Debido al ruido introducido, los resultados no concuerdan en porcentaje de los estados obtenidos pero, además, el sistema real introduce porcentajes en estados que no deberían ni aparecer, ya que el circuito original es de tres qubits.

A causa del ruido, estos estados tienen probabilidades de aparecer en el sistema real, por lo que se aprecia que el ruido puede llegar a ser un gran inconveniente a la hora de diseñar circuitos cuánticos.

3.4. Flujo de ejecución del diseño a implementar

En este apartado se va a mostrar y comentar el flujo de ejecución diseñado para la implementación del programa final, de esta forma, se comprenderá claramente el objetivo final que tienen los pasos realizados a lo largo de todo el desarrollo.

Se utilizarán colores para distinguir entre las partes que se deben desarrollar de las partes que no es necesario, ya que simplemente habrá que implementarlos en el programa final. En verde, los pasos a desarrollar, en rojo, los pasos que no es necesario desarrollarlos. Además, se acompañará cada uno de los pasos con el programa o la librería que será utilizada para ello. De esta forma se puede comprender con un solo vistazo las necesidades que requiere el desarrollo del programa final completo.

La explicación detalla de cada una de las tareas se realizará en los capítulos próximos, mientras que en este apartado se aportará una visión global de los pasos que seguirá la ejecución del programa final. En todas las ejecuciones se partirá de una serie de datos de activos previamente recopilados. Además durante las pruebas se realizarán variaciones con distintos activos para comprobar el correcto funcionamiento del diseño.

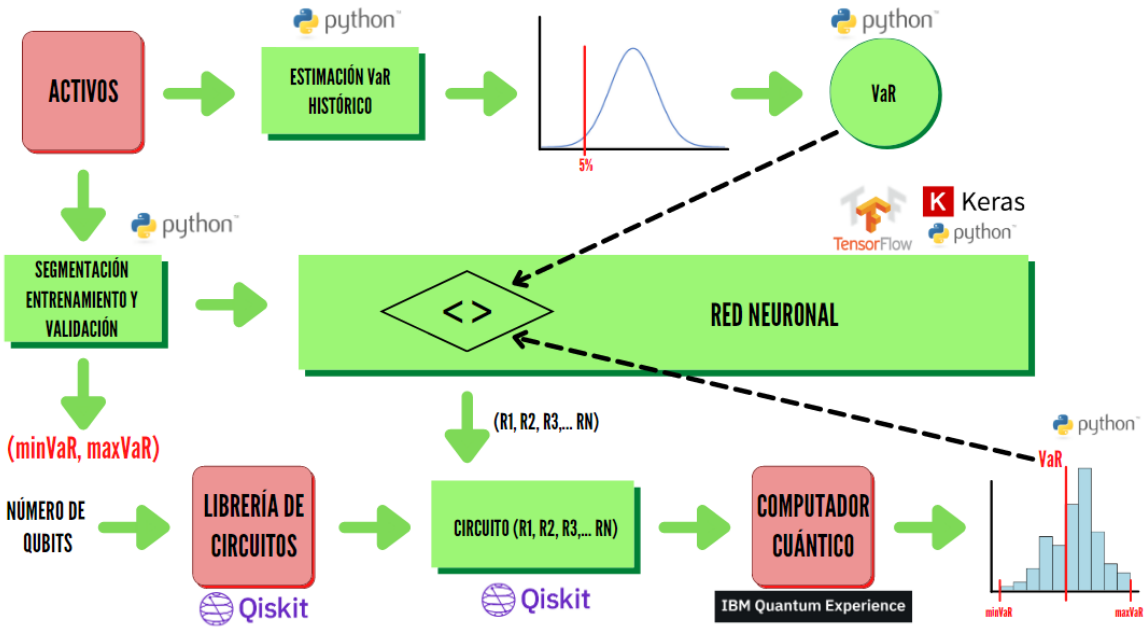

Figura 3.9: Flujo de ejecución del programa final.

En primer lugar, desde los activos iniciales se realizan una serie de pasos previos, comenzando con la estimación del VaR histórico. En esta estimación se obtiene el VaR histórico, que servirá para realizar cálculos del coste en la red neuronal, pero antes de poder realizar la comparación es necesaria la agrupación de todos los activos en grupos del número deseado, que por defecto serán agrupaciones de 20 en 20. Una vez se tienen todas estas agrupaciones, es necesario el cálculo del corte del 5 % del histograma asociado de cada una de las agrupaciones, de forma que se obtenga el VaR histórico final de cada una de las agrupaciones.

El siguiente paso necesario es la segmentación de los datos iniciales en datos de entrenamiento y validación, en el caso de que se tenga una cantidad de datos iniciales baja, se utilizará el método de establecer como entrenamiento todos los datos menos uno, que servirá de validación del sistema. De este paso también se extraerá el valor del VaR mínimo y máximo, que se utilizarán en pasos siguientes.

Una vez que se tienen todos los datos segmentados, estos se introducen a la red neuronal para que pueda comenzar el aprendizaje. La red neuronal realizará los cálculos necesarios para obtener las rotaciones necesarias del circuito cuántico. Este circuito genérico será obtenido de la librería de circuitos Qiskit con la cantidad de qubits que se desea utilizar. Al obtener el circuito, se editará con las rotaciones de los qubits obtenidas de la red neuronal.

Posteriormente, cada uno de los circuitos resultantes serán ejecutados en el computador cuántico seleccionado y se obtendrá el VaR calculado mediante la utilización del rango obtenido entre el VaR mínimo y máximo. La selección del punto de corte de este nuevo VaR será decidida en los capítulos posteriores.

Con el VaR predicho por la ejecución del circuito cuántico editado de la red neuronal y el VaR histórico calculado de la estimación del VaR histórico, se procederá a calcular el coste y realizar los ajustes necesarios en las rotaciones de los qubits para habilitar el aprendizaje. De esta forma, con la comparación entre los resultados de los mismos, con una distancia (por ejemplo, la distancia euclídea), se calculará el coste de las iteraciones y de las distintas épocas.

El proceso total del flujo de ejecución sería, se introducen a la red neuronal los valores iniciales de los activos y sus respectivos VaR históricos calculados (por agrupaciones de 20). Estas agrupaciones van realizando el proceso anterior (lectura, obtención de las rotaciones, ejecución

en el sistema cuántico y obtención del resultado del VaR estimado), hasta que se completa la primera época. La red neuronal altera el valor de los coeficioentes de las neuronas, y comienza la nueva época, donde vuelve a ocurrir el mismo proceso con los nuevos coeficientes. De esta forma, realizando una serie de época, la red neuraonal es capaz de ir reduciendo el coste y por tanto, de ir aprendiendo a realizar la tarea.

4

Desarrollo

4.1. Plan de trabajo

En este capítulo se va a exponer las ideas iniciales y los procedimientos seguidos en cada uno de los pasos realizados a lo largo del TFM. También se van a comentar las conclusiones obtenidas de esta serie de tareas.

Para la correcta comprensión de las tareas y los pasos realizados, se ha decidido comentarlas cronológicamente, para exponer las dudas y los problemas que fueron surgiendo a lo largo de su realización. Además, se han seleccionado las tareas más destacables en la ejecución y que aportaron un beneficio al desarrollo del mismo (ya sea por su correcto funcionamiento o por los conocimientos adquiridos en ellas).

Pero antes de adentrarse en la explicación de cada una de las tareas, se verá un esquema del flujo de desarrollo del TFM, que puede servir para ilustrar el estado del proyecto, en cada uno de los siguientes apartados de esta memoria, permitiendo volver a la figura en cualquier momento y observar el punto en el que se encuentra cada apartado o capítulo.

Se va a continuar con la leyenda de colores establecida, de manera que en color verde, se encontrarán los pasos que necesitan ser desarrollados, mientras que en rojo se encontrarán los pasos que no necesitan desarrollo previo. Las explicaciones de cada uno de estos pasos se expondrán a lo largo de este mismo capítulo y de los capítulos sucesivos.

Esta figura tiene la función de facilitar la comprensión global del proyecto. Si se observa comenzando por la esquina superior izquierda, se puede seguir el flujo del desarrollo del TFM, siendo las flechas en negrita el flujo seguido y las flechas de puntos las aportaciones necesarias de un paso a otro. Por último, el color amarillo ha sido utilizado para identificar las pruebas realizadas para la obtención de los resultados.

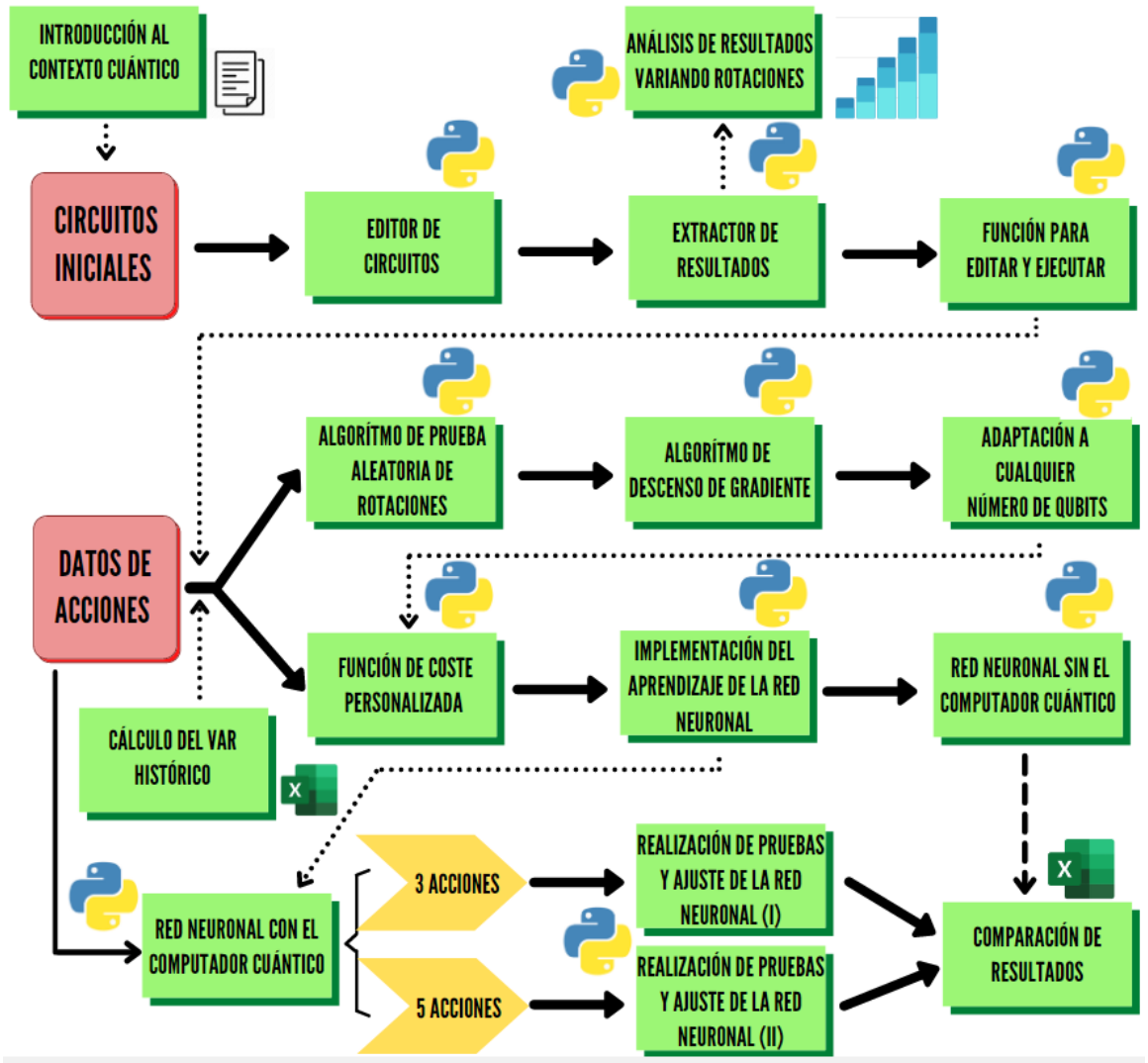

Figura 4.1: Flujo de desarrollo del TFM.

4.2. Desarrollo de los primeros programas y circuitos

En este breve apartado se van a comentar los primeros desarrollos en la realización de este TFM. Como se ha comentado en el capítulo anterior, mediante la utilización de las herramientas proporcionadas por la plataforma de diseño de circuitos cuánticos de IBM, se logró ejecutar los primeros circuitos creados, de forma que se podía realizar el diseño, la ejecución y la revisión de los resultados para realizar ajustes en el circuito.

El siguiente paso con el que se continuo fue intentar realizar los primeros diseños y ejecuciones en un programa, de forma que se pudiesen diseñar y ejecutar automáticamente, sin necesidad de situar los bloques a mano, establecer los valores uno a uno, seleccionar dónde se ejecutaban y mandarlos ejecutar de forma manual.

Para lograr realizar este programa inicial, del que se partiría para la realización de los siguientes pasos, se decidió utilizar la herramienta de programación de la plataforma de IBM, basada en *Jupyter* del lenguaje Python.

En esta herramienta se utilizó Python para crear el programa que, mediante la utilización de la librería de Qiskit, permitió la instanciación de los primeros circuitos cuánticos en lenguaje

de programación. De esta forma, se podía realizar el diseño del circuito, con los valores de las rotaciones de los qubits por defecto, para posteriormente, ser mandados a ejecutar.

Una vez se comprobó el correcto funcionamiento de la instanciación en el programa de Python y que la posibilidad de ejecutar estas acciones era real en la plataforma de IBM, se procedió a realizar los siguientes pasos necesarios para el desarrollo del TFM.

4.3. Desarrollo y análisis de circuitos personalizados

En esta sección se va a comentar el desarrollo de los circuitos personalizados, es decir, circuitos con las rotaciones editadas de los circuitos iniciales, permitiendo así, variar los valores para realizar pruebas y continuar los pasos previstos.

4.3.1. Introduccion

Para ser capaces de lograr crear un programa que automatice los principales pasos para el diseño y ejecución, se necesita tener funciones que sean capaces de editar los circuitos iniciales y que sean capaces de extraer los resultados después de la ejecución. Se ha divido la creación de estas funciones habilitantes para facilitar su posterior explicación.

4.3.2. Desarrollo del editor de circuitos

En el comienzo de la ejecución del programa, lo primero que se necesita es la edición del circuito inicial generado con la librería Qiskit. Para realizar esta personalización del circuito se necesita poder editar de alguna forma el circuito.

El circuito cuántico, como ya se ha comentado, se genera en lenguaje Qiskit, siendo difícil su edición, pero, mediante la librería Qiskit, se puede realizar una transformación del circuito en Qiskit al circuito en lenguaje QASM. Esta forma de presentar el circuito permite su edición como si se tratase de texto plano, teniendo para cada uno de los elementos un formato específico.

Una vez se ha transformado el circuito al lenguaje QASM, se buscan los bloques de rotaciones de los qubits para personalizar cada una de sus rotaciones, permitiendo no variarla en el caso de que así se desee. Esta capacidad de edición también necesita establecer la barrera para finalizar las operaciones computacionales y los medidores situados en cada uno de los qubits, dado que el circuito inicial en formato Qiskit no tiene establecida la medición de los qubits.

Además de realizar estos cambios, la función debe tener la capacidad de funcionar correctamente si solo se pasan unas pocas rotaciones, porque no se pretende cambiar todo el circuito, o eliminar el exceso de rotaciones en el caso de que el diseñador haya pasado demasiados valores.

Por último, esta función necesita pasar el circuito personalizado al formato Qiskit de nuevo para su posterior ejecución, ya que todos las ejecuciones y visionados se realizan en ese formato.

Una de las mayores dificultades a la hora de realizar esta función fue la diferencia existente entre las distintas versiones de Qiskit, que van variando el nombre de los bloques utilizados y fuerzan la actualización de la función de forma continua a lo largo del TFM. También se encontraron dificultades para encontrar la forma óptima de realizar estos cambios en las rotaciones y para generar un circuito cuántico correcto que pueda ser ejecutado y aporte resultados que se puedan examinar, comprender y utilizar en el futuro.

Para apreciar las diferencias entre el circuito en formato Qiskit y el circuito en formato QASM, se ha decidido incluir en este apartado el mismo circuito cuántico en ambos lenguajes.

Circuito en qiskit

```
1  from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2  from numpy import pi
3
4  qreg_q = QuantumRegister(5, 'q')
5  creg_c = ClassicalRegister(5, 'c')
6  circuit = QuantumCircuit(qreg_q, creg_c)
7
8  circuit.u3(1.98, 0, 0, qreg_q[2])
9  circuit.cx(qreg_q[2], qreg_q[1])
10 circuit.u3(0.0397, 0, 0, qreg_q[1])
11 circuit.cx(qreg_q[2], qreg_q[0])
12 circuit.u3(0.203, 0, 0, qreg_q[0])
13 circuit.cx(qreg_q[1], qreg_q[0])
14 circuit.cx(qreg_q[2], qreg_q[0])
15 circuit.cx(qreg_q[1], qreg_q[0])
16 circuit.u3(0.101, 0, 0, qreg_q[0])
17 circuit.cx(qreg_q[1], qreg_q[0])
18 circuit.u3(1.68, 0, 0, qreg_q[0])
19 circuit.barrier(qreg_q[0], qreg_q[1], qreg_q[2], qreg_q[3], qreg_q[4])
20 circuit.measure(qreg_q[0], creg_c[0])
21 circuit.measure(qreg_q[1], creg_c[1])
22 circuit.measure(qreg_q[2], creg_c[2])
23 circuit.measure(qreg_q[3], creg_c[3])
24 circuit.measure(qreg_q[4], creg_c[4])
```

Circuito en QASM

```
1  OPENQASM 2.0;
2  include "qelib1.inc";
3
4  qreg q[5];
5  creg c[5];
6
7  u3(1.98,0,0) q[2];
8  cx q[2],q[1];
9  u3(0.0397,0,0) q[1];
10 cx q[2],q[0];
11 u3(0.203,0,0) q[0];
12 cx q[1],q[0];
13 cx q[2],q[0];
14 cx q[1],q[0];
15 u3(0.101,0,0) q[0];
16 cx q[1],q[0];
17 u3(1.68,0,0) q[0];
18 barrier q[0],q[1],q[2],q[3],q[4];
19 measure q[0] -> c[0];
20 measure q[1] -> c[1];
21 measure q[2] -> c[2];
22 measure q[3] -> c[3];
23 measure q[4] -> c[4];
```

Figura 4.2: Comparación de un circuito en Qiskit y en QASM.

Se pueden apreciar las notables diferencias entre ambos lenguajes, mientras el circuito en formato Qiskit tiene una apariencia más de lenguaje de programación, el lenguaje QASM tiene un aspecto poco amigable para ser leído por una persona. Pero, aunque sea un aspecto menos amigable, el problema de intentar editar el circuito en el formato original es que, como se puede observar, todo el circuito Qiskit se encuentra contenido en una variable creada inicialmente "circuito" en la línea seis. Por lo que se van añadiendo cada uno de los bloques elegidos uno a uno de forma cronológica y editar uno en concreto se hace un proceso demasiado tedioso.

Una de las posibilidades para realizarlo en el formato original sería intentar rehacer el circuito por completo desde cero, pero esto resulta muy poco eficiente, por lo que transformarlo para ser editado y devolverlo a su lenguaje original una vez personalizado resulta la mejor opción (y la que se ha llevado a cabo).

4.3.3. Desarrollo del extractor de resultados

Mediante la función desarrollada previamente, ya se podría editar un circuito cuántico para asignarle los valores de rotación de los qubits deseados. Ahora, el siguiente paso sería el de ejecutar y extraer los resultados obtenidos para poder realizar operaciones con ellos hasta alcanzar un resultado que se consideraría el resultado de la ejecución final.

Para poder realizar esta obtención de un resultado final desde un histograma proporcionado por el sistema cuántico, es necesario un proceso sistemático y bien meditado para encontrar el mejor de los posibles. Se podría considerar que las probabilidades que aparecen en el histograma del resultado del computador son, en sí mismas, los resultados de las probabilidades de los escalones del riesgo.

Esto se puede explicar de esta forma, si en una serie de valores correlacionados se encuentra que, estos pueden ser modelados por una gaussiana cada uno de ellos, de forma independiente, entonces es posible, por las propiedades de las gaussianas, encontrar una gaussiana que sea capaz de modelar aproximadamente al conjunto correlacionado de los valores.

De esta forma, si se parte de una situación dónde se tienen las gaussianas, se puede modelar la gaussiana aproximada y conocer sus propiedades. Con esta gaussiana aproximada se puede generar el eje x de la representación final, dividiendo los valores por probabilidad dependiendo de la cantidad de estados existentes en la aproximación. Si se da el caso en que se pretende realizar una simulación con cuatro qubits, entonces se dividen los posibles valores de la gaussiana en la cantidad de estados que estos generan (en este caso serían 16), asignando a cada partición la misma probabilidad, por lo que este eje x no aumentaría siempre en la misma cantidad, sino que iría creciendo la diferencia hacia los bordes.

Con este eje x generado, es posible entonces realizar un corte en el histograma resultante de la ejecución en su 5 % para obtener el riesgo deseado y, posteriormente, observar en qué estado ha caído ese punto y encontrar el valor real obtenido mediante la aproximación de la gaussiana con el eje generado.

Este método permite pasar de un valor probabilístico en forma de histograma, a un valor real que se puede utilizar a continuación, en cualquier posible operación.

4.3.4. Desarrollo de la función para editar y ejecutar

Teniendo las funciones de edición y obtención de los resultados, la parte que falta es la ejecución automática de los circuitos cuánticos diseñados, que además actua como la parte necesaria para enlazar ambas funciones creadas previamente.

Para la creación de este enlace entre ambas funciones que serviría para ir automatizando el proceso de creación y obtención de resultados, se necesita primero realizar la selección del computador cuántico en el que se va a ejecutar el circuito, o en cambio, si se realizará en un simulador. La selección de este computador, se utilizan las funciones de Qiskit para elegir el sistema disponible con las características necesarias para la ejecución, con la menor espera posible.

Una vez que se tiene el sistema en el que se va a realizar la ejecución, el siguiente paso es el envío de la ejecución y la espera hasta la recepción del histograma solución. Al recibir ese histograma con las probabilidades de cada uno de los estados, se pasa a un formato que pueda ser legible por la función de extracción de los resultados, para convertirlo al valor final que será utilizado.

4.3.5. Conclusiones de esta parte del desarrollo

Con la realización de este apartado del desarrollo, se ha comenzado a comprender las necesidades que aparecerán al realizar la implementación del programa final. Para el correcto funcionamiento del programa se podrán utilizar las funciones creadas. o bien actualizarlas para asegurar que funcione correctamente el conjunto del programa.

En este punto se ha observado la dificultad de trabajar utilizando circuitos cuánticos para realizar ejecuciones complejas. Al no conocer de qué forma afectan las rotaciones de los qubits cuando se pasa a ejecutar circuitos complejos, es bastante complejo plantearse diseñar circuitos correctamente para realizar ejecuciones precisas.

Mediante esta serie de funciones, se profundiza más en el TFM y se comprueba la capacidad que existe para trabajar con circuitos cuánticos. Todas las tareas realizadas van aportando

conocimiento que será muy útil a la hora de realizar la implementación final y permiten la aparición de nuevas dudas y hechos que hay que comprobar e intentar resolver para avanzar hacia el paso final, donde todo el conocimiento adquirido será de una utilidad inmensa.

4.4. Implementación de algoritmos de I.A. simples para obtener el circuito ideal para el resultado deseado

En esta sección del desarrollo, se va a exponer una serie de tareas realizadas para profundizar en el conocimiento del comportamiento de los circuitos cuánticos, a los que se les altera el valor de las rotaciones de los qubits.

4.4.1. Introduccion

Para poder obtener más conocimiento sobre las rotaciones de los qubits y empezar a comprender la elección de las mismas se pensaron dos tareas. Una tarea dependía directamente de la otra dado que se trataría de un algoritmo que mejoraría el paso anterior.

Inicialmente, para comprender estas rotaciones, se pensó en realizar un algoritmo que implementase un método de fuerza bruta, dónde las rotaciones se eligen de forma aleatoria hasta alcanzar un mínimo de distancia con el valor objetivo. Y posteriormente este sería mejorado por un algoritmo de descenso de gradiente. La principal razón para realizar estos pasos era observar si un mismo circuito, con distintas rotaciones, era capaz de generar resultados que fueran variando en cada caso. Y cuánto eran capaces de variar los circuitos en estas pruebas.

Se va a pasar a exponer la realización de estos pasos, con los problemas encontrados y los conocimientos adquiridos en ellos.

4.4.2. Implementación de un algoritmo de prueba aleatoria de rotaciones (fuerza bruta)

Antes de pasar a comentar este paso, es necesario tener en mente el significado de este método para obtener resultados. Cuando se está intentado resolver un problema en el que no se sabe con precisión cómo deben ser los valores introducidos para lograr el objetivo final (en este caso, que el diseño genere un VaR establecido), suele ser una buena práctica realizar un algoritmo sencillo de prueba con números aleatorios.

Estos algoritmos proporcionan la capacidad de ser ejecutados y obtener un resultado, además, con el paso de las iteraciones, se puede adquirir conocimiento para resolver el mismo problema de una forma más eficiente. De forma que el algoritmo genera valores aleatorios para probar una y otra vez hasta alcanzar cierto umbral establecido, donde se detiene y arroja el resultado final.

El primer paso del aprendizaje sobre las rotaciones de los qubits para alcanzar resultados deseados fue el de realizar un algoritmo de fuerza bruta que permitiese obtener las rotaciones óptimas para cada valor. Para poder realizar este algoritmo, es necesaria la implementación del mismo con los pasos desarrollados previamente, pues, es necesario el diseño del circuito en cada una de las ejecuciones con los valores generados.

Para generar una serie de valores independientes, dependiendo de la cantidad de rotaciones que se prendía probar, se fue variando la cantidad de rotaciones que se modificaban en las distintas ejecuciones. Por ejemplo, en la primera prueba, se cambiaba tan solo el primer qubits de los que estaban disponibles, de forma que se podía observar si el cambio de un qubit era capaz de variar la ejecución por completo.

De esta forma, se podía ir comprobando paso a paso la capacidad que se conseguía frente a la cantidad de qubits con las rotaciones cambiadas. Como último paso, se optó por otorgarle al algoritmo la capacidad de variar todas las rotaciones de los qubits a su elección, por lo que el algoritmo tenía la capacidad teórica de llegar, sino al número exacto, al estado más cercano posible alcanzable.

Pero, como se puede pensar, obtener las rotaciones de forma aleatoria, no resulta ser la forma más eficiente de alcanzar el mínimo absoluto de distancia frente a un punto establecido previamente, sino que, por el contrario, teóricamente, puede realizar infinitas iteraciones para alcanzar el punto deseado. Debido a esto, el siguiente paso lógico es implementar un algoritmo más complejo que permita realizar estas pruebas de forma más rápida y correcta.

4.4.3. Implementación de un algoritmo del tipo descenso por gradiente

Los algoritmos de descenso por gradiente son altamente utilizados en la actualidad, Esto se debe a que proporcionan unas grandes capacidades para encontrar mínimos absolutos de una función. Estos algoritmos comienzan sus iteraciones situándose en un punto aleatorio de la función, es decir, la primera iteración se realiza con números aleatorios.

Una vez se ha situado el inicio del descenso en un punto de la función de los posibles resultados, el algoritmo intenta desplazarse hacia un lado de la función una distancia establecida previamente, si el resultado empeora (hablando en la distancia al resultado deseado), se intenta desplazar hacia el lado contrario. Este proceso se repite hasta que el desplazamiento hacia ambos lados produce un empeoramiento, por lo que se ha encontrado un mínimo.

El algoritmo debe iniciarse en varios puntos de la función para asegurarse de que se está encontrando un mínimo absoluto y no se ha quedado estancado en un mínimo local. Además, es necesario variar el parámetro que decide la distancia a la que se desplaza para un lado o el otro, al cambiarse este valor, se pueden escapar de pequeñas variaciones en la función si tenemos un parámetro demasiado pequeño o no salirse de mínimos muy abruptos escalando por las paredes, en el caso de un parámetro demasiado grande.

Como se puede apreciar, para realizar este tipo de algoritmos es necesario realizar una serie de pruebas y ajustes para asegurarse de que está funcionando correctamente. Pero una vez se tiene el algoritmo funcionando, resulta ser un algoritmo bastante visual si se decide imprimir el resultado y las acciones tomadas, además de ser uno de los mejores algoritmos para alcanzar mínimos absolutos.

Para la realización de este algoritmo con las funciones previas, se sustituyó el algoritmo de fuerza bruta por el nuevo de descenso por gradiente. Pero claro, el descenso por gradiente suele utilizarse cuando se conocen los cambios que se deben realizar para avanzar hacia un lado de la función o el lado contrario. Por lo que, en el caso del circuito cuántico, es necesario pensarlo primero detenidamente.

Se llegó a la conclusión de que al ser rotaciones de los qubits y variar estas entre 0 y 2π, se podía asumir que estos movimientos hacia un lado de la función y el otro, se podian asignar a una rotación a la derecha o a la izquierda. Por lo que el algoritmo se podía implementar con variaciones en los qubits para alcanzar el mínimo absoluto de la función distancia al resultado.

Pero, ¿se debería ir variando uno a uno o todos a la vez? Esta duda fue resulta de la única forma posible, a base de pruebas y de la observación de resultado. No se podía conocer esta respuesta previamente porque el algoritmo anterior no variaba, sino que generaba puntos aleatorios de la distribución encontrando el menor posible. Por lo que inicialmente se variaron todos al mismo tiempo con el mismo parámetro de desplazamiento.

Se pudo comprobar que desplazando todos a la vez se producían efectos no deseados, y además, no era posible encontrar una lógica en el movimiento coherente con el algoritmo desarrollado. Por lo que se optó por ir desplazando cada rotación una a una, como si se tratase de funciones correlacionadas que juntas forman la función de distancia total.

Desplazando cada uno de los qubits se podía alcanzar mínimos, hasta que el algoritmo se detenía y pasaba al siguiente qubit, de esta forma se alcanzaba un mínimo. Una vez se habían ajustado todos los qubits, se procedía a volver a realizar el mismo proceso desde el inicio, por si la rotación de los qubits anteriores pudiese aumentar la precisión, estableciendo una repetición en el algoritmo a modo de depuración del resultado.

Una vez se desarrolló el algoritmo correctamente se pudieron pasar a realizar pruebas, para comprobar que era posible acercarse al resultado objetivo, pero que la resolución de los resultados no alcanza para obtener los valores exactos esperados. Al establecer una cantidad de qubits, intrínsecamente, se está definiendo la resolución del problema, siendo posible un movimiento discreto, pero no uno con valores reales.

4.4.4. Prueba y adaptación de los algoritmos a distinto número de qubits

Teniendo ambos algoritmos desarrollados, era necesario observar su funcionamiento cuando se producía un escalado en la cantidad de qubits proporcionados. Las funciones base habían sido previamente preparadas para estos cambios y podían variarse sin ningún problema, pero los algoritmos habían sido desarrollados, en primera instancia, como pruebas sobre un circuito con un número de qubits específico.

Primero se comenzó variando el algoritmo de fuerza bruta, donde se ajustó la cantidad de qubits que serían cambiados en cada iteración. Para encontrar la cantidad de puertas en las que se producen rotaciones de los electrones de los circuitos, era necesario observar la cantidad que posee cada uno de los circuitos iniciales que se pueden generar con la librería Qiskit. Después de una serie de pruebas, se pudo comprobar que los circuitos cuánticos estaban compuestos por 2^x puertas con rotaciones, donde x es la cantidad de qubits establecida en la creación del circuito cuántico.

Una vez vista esta relación, lo que se necesitaba era que la cantidad de qubits que se variaban en cada uno de los algoritmos anteriores fuese variable, teniendo en cuenta la cantidad de qubits utilizados en cada una de las pruebas realizadas. Al establecer esta cantidad variable, se podían realizar pruebas con distinta cantidad de qubits de forma automática, sin necesidad de preocuparse de la cantidad de rotaciones del circuito o cantidad de qubits disponibles.

4.4.5. Conclusiones de esta parte del desarrollo

Gracias a estos pasos desarrollados se pudo aprender más de las rotaciones en cada uno de los circuitos, y observar la forma en la que afectan al resultado final. Principalmente, el paso del algoritmo de fuerza bruta sirvió de ayuda para realizar el que verdaderamente aportaría beneficios al aprendizaje, el algoritmo de descenso por gradiente.

Al utilizar ambos métodos para realizar cálculos del VaR para acercarlo a un valor específico, se pudo observar que los resultados que se pueden obtener, son discretos, dependiendo la cantidad de tramos disponibles de la cantidad de estados en los que se pueden situar los qubits establecidos. Confirmando las sospechas establecidas sobre el caso que íbamos a probar.

Una vez se comprobaron ciertos conocimientos necesarios sobre las rotaciones de los qubits, se pudo plantear avanzar en el proceso para alcanzar el programa final, donde todo el conocimiento

adquirido en pasos anteriores sería muy valioso. Es mucho más sencillo crear un programa sobre algo que se conoce, que crear un programa sobre un problema que no se comprende en absoluto.

Teniendo este conocimiento, se puede discernir entre, si algo no se está haciendo correctamente, o funciona de forma óptima, e incluso, en última instancia, entender si el problema a solucionar se puede alcanzar con ese planteamiento o es inviable.

4.5. Implementación de redes neuronales para realizar cálculos

En esta sección se van a exponer los pasos que tienen relación con la aplicación de redes neuronales. Hay que destacar que todos los pasos previos al programa final tienen la finalidad de facilitar la realización del mismo, mientras que se pueden aplicar ciertas variaciones en el programa final que serán explicadas, con sus motivos y resultados.

Además, esta sección servirá de iniciación al capítulo siguiente, ya que la mayor parte del desarrollo llevado a cabo en esta sección se utilizará para realizar el programa final, realizando los cambios y ajustes que se consideren necesarios.

4.5.1. Introduccion

Con todo el conocimiento adquirido con los pasos anteriores y con las funciones desarrolladas, se puede continuar con el progreso previsto, comenzando a realizar los pasos relacionados con las redes neuronales.

Estos pasos resultan ser de gran utilidad porque están relacionados directamente con el programa final, que se basará en estos para su creación. La red neuronal resulta ser una herramienta ideal para encontrar la solución a problemas en los que se tienen una serie de valores iniciales y resultado final. Con estos resultados se puede entrenar la red neuronal, que permite resolver problemas en los que la solución no resulta trivial.

Pero al utilizar una red neuronal, se encuentran una serie de problemas que deben ser solucionados antes de comenzar a funcionar. Todos los problemas encontrados se comentarán y se planteará su solución a lo largo de este apartado y el siguiente capítulo.

4.5.2. Desarrollo de la función de coste que recibe las rotaciones

El principal problema que se encuentra para implementar la red neuronal, en el cálculo de los circuitos cuánticos, es la necesidad de crear una función de cálculo de coste propia. Este problema se debe a que, en un entrenamiento normal de una red neuronal, los resultados que calcula la propia red son comparables con el dato correcto que se le proporciona. Pero esto no ocurre así en este caso.

La red neuronal va a ser utilizada para que aprenda sobre las rotaciones de los qubits y sea capaz de calcular las rotaciones necesarias para alcanzar el VaR deseado, es decir, la red neuronal no va a sacar un resultado concreto que pueda ser comparable con el proporcionado, sino que, va a obtener las rotaciones que serán ejecutadas en el computador cuántico para alcanzar un resultado que se pueda comparar.

De esta forma, se necesita establecer un paso previo al cálculo del coste de las iteraciones, donde se calculen los valores del VaR generados que pueden ser comparados con el valor suministrado a la red neuronal. Este paso previo debe ser incluido en el cálculo de la función de coste para que toda la red neuronal funcione correctamente.

El problema de la función de coste va a ser ilustrado para una mayor claridad, ya que este paso resulta ser de crucial importancia a la hora de utilizar la red neuronal. También, se va a realizar una pequeña explicación del planteamiento para el paso final y para demostrar esta necesidad implícita que tiene su desarrollo. La función de coste resultará ser una parte diferenciadora de otras redes neuronales, dónde se utilizan funciones de coste predeterminadas o al menos, que no efectúan cambios en las dimensiones del resultado.

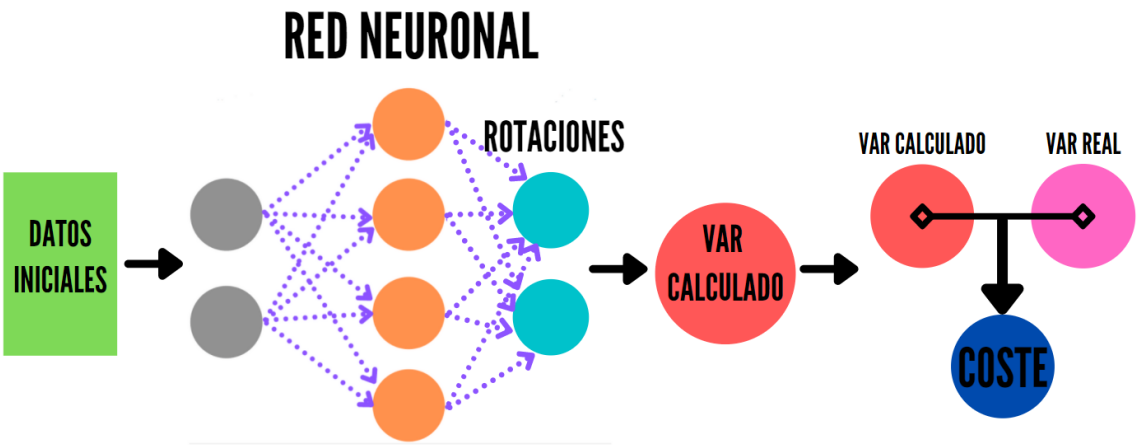

Figura 4.3: Esquema de la red neuronal con el computador cuántico.

En la figura se pueden observar los pasos que se siguen para realizar la implementación de la red neuronal junto al computador cuántico, esta figura será explicada en profundidad en el diseño del programa final. Por ahora, la explicación se centrará en el paso del cálculo del coste. Para llegar al coste, es necesario que la red neuronal reciba los datos de entrada, que estos atraviesen las neuronas con sus valores establecidos y la red nos proporcione los resultados de las rotaciones necesarias para la ejecución.

Una vez calculadas las rotaciones, es necesario que la función de coste realice el cálculo del VaR correspondiente, puesto que sino no se pueden comparar los resultados y hallar el coste. Para esto se ha realizado el cálculo en el interior de la función de coste del VaR resultante de cada iteración.

Esta función de coste personalizada ha tenido bastantes problemas en su realización, desde el problema de manejar las dimensiones para que todo funcione correctamente, hasta un problema derivado del uso de la librería de Python *keras*, que necesita un procedimiento específico a la hora de definir las funciones de coste personalizadas, manteniendo las variables que se proporcionan a la función de coste sin ningún cambio para que el modelo no se "pierda".

Por último, hay que destacar que se ha decidido utilizar una distancia euclídea para calcular el coste final con ambos valores del VaR, puesto que se quiere premiar de gran forma la aproximación a los resultados proporcionados.

4.5.3. Implementación del entrenamiento de la red neuronal

El entrenamiento de la red neuronal se realizará mediante el uso de la función de coste creada anteriormente, con esta función se puede intentar reducir el coste hasta el punto deseado, habilitando de esta forma el aprendizaje sobre la tarea deseada.

Para realizar el ajuste de las capas y de la cantidad de neuronas que necesita el entrenamiento, se fueron realizando pruebas para ir ajustando el modelo a medida que se ejecutaba. Para el

entrenamiento hay que definir tres grandes conceptos, la cantidad de capas, la cantidad de neuronas en cada capa y la activación de cada capa.

La cantidad de capas comenzará siendo de dos, aunque podrá aumentar si se considera necesario, se piensa que tres capas serían más que suficientes. La cantidad de neuronas irá variando en función del resultado de las ejecuciones y comenzará con un valor relativamente elevado. De todas formas, el valor de la cantidad de neuronas de cada capa se irá ajustando a medida que se realice el programa final y se vayan observando los resultados.

En la selección de la activación se encuentra la decisión más importante. Se necesita una activación que varíe en un rango cercano o mayor que las rotaciones de los qubits, es decir, 2π. Por esta razón, hay que utilizar una serie de posibles activaciones que generan este rango de valores en las neuronas, evitando utilizar funciones como seno o coseno donde los valores van desde -1 a 1.

Una vez se tienen estas tres características definidas, se puede comenzar a desarrollar el entrenamiento mediante la librería keras y programar toda la red neuronal para el correcto funcionamiento. Después, se procede a realizar cambios y ajustes para observar el funcionamiento e intentar mejorar el resultado todo lo posible.

4.5.4. Desarrollo de una red neuronal saltándose el computador cuántico para comparar resultados

Mientras se estaban desarrollando los pasos necesarios para la realización de la red neuronal, se pensó en la necesidad de la creación de otra red neuronal diferente para que se pudiera comparar con la creada para el computador cuántico.

Dada esta necesidad, se comenzó con el desarrollo de una red neuronal que hiciese la función de comparador con el que se pueda evaluar el rendimiento obtenido y si se ha sido capaz de mejorar este baremo, o en cambio, la red neuronal sin el computador cuántico funciona mejor que el que utiliza el computador. Para explicar de forma clara esta parte, se va a optar por utilizar una figura similar a la anterior, de forma que el esquema vaya siendo familiar.

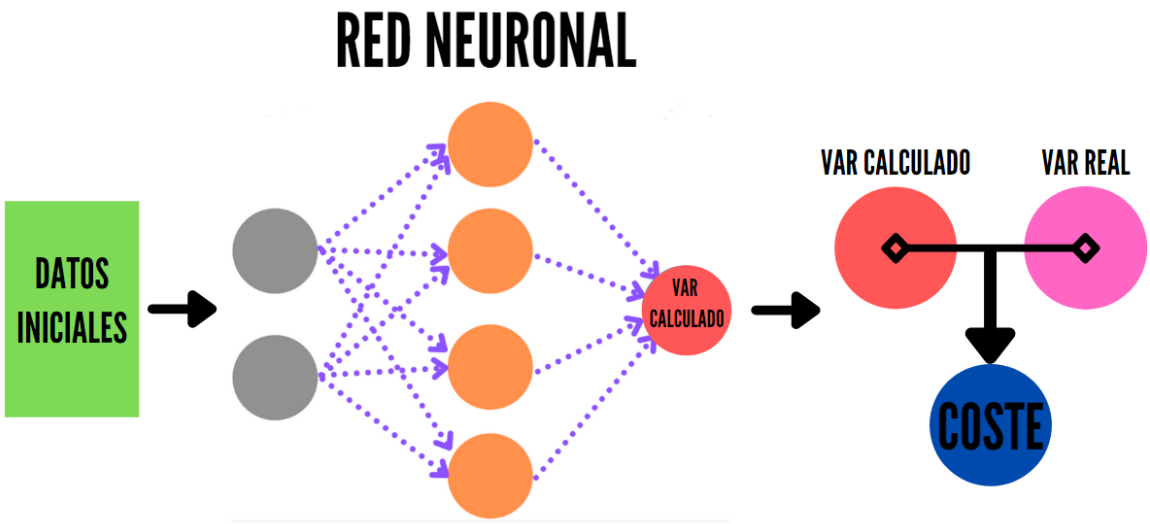

Figura 4.4: Esquema de la red neuronal sin el computador cuántico.

Como se puede apreciar en la figura, en el caso de la red neuronal sin utilizar el computador cuántico, el resultado que otorga la red es directamente el VaR calculado. Debido a esto, no es

necesario la utilización de funciones de coste complejas, simplemente es necesaria la decisión de qué distancia se utiliza para el cálculo.

Esta red recibirá los datos sin ningún tratamiento previo y sin ser ordenados. Solo recibirá el dato de cada día, del valor que conste en el cálculo del VaR correspondiente. De estos datos, deberá extraer sus características necesarias para calcular la tarea requerida. Para el correcto entrenamiento, se suministrará también los valores del VaR esperados a la salida, de forma que se tienen los valores de entrada y de salida de cada iteración, siendo un esquema de aprendizaje supervisado.

En esta red habrá que tomar decisiones de diseño al igual que en el desarrollo del programa final, por lo que las decisiones tomadas, serán expuestas al comparar los resultados, de forma que haya mayor claridad en el momento de apreciar estos resultados.

4.5.5. Conclusiones de esta parte del desarrollo

Con la finalización del desarrollo de los pasos relacionados con las redes neuronales, se pudo comprobar las posibilidades que van a otorgar estas, y la capacidad de personalización que tienen. La función de coste desarrollada servirá para poder utilizar el computador cuántico para los cálculos del VaR resultante.

Además, con la creación del esquema independiente del computador cuántico se habilita la comparación frente a otro sistema. Es importante poder obtener una comparación a la hora de observar los resultados, ya que, un sistema que funcione mejor o peor que otros otorgara un punto de referencia excelente.

4.6. Conclusiones del desarrollo

A lo largo de la realización de cada uno de los pasos se ha ido adquiriendo el conocimiento necesario para afrontar el desarrollo del programa final. En este programa se implementarán una gran parte de las funciones desarrolladas y se ajustará cada una de ellas para su funcionamiento conjunto de manera correcta.

Las funciones desarrolladas inicialmente han servido de bases para el desarrollo de los siguientes pasos, en todos los programas es necesaria la personalización de los circuitos cuánticos y la obtención de los resultados de forma correcta. También es de gran utilidad poder variar la cantidad de qubits que se utilizan en cada una de las ejecuciones, para poder observar los cambios que van sucediendo con más qubits o menos.

Los algoritmos capaces de obtener las rotaciones necesarias para las ejecuciones han servido para familiarizarse con las rotaciones de los mismos, para contrastar en qué rango se encontraban estas rotaciones y observar la cantidad de rotaciones existentes en los circuitos cuánticos iniciales de la librería Qiskit. También han permitido conocer el funcionamiento de estos algoritmos clásicos para encontrar mínimos en un computador cuántico.

Por último, los pasos relacionados con las redes neuronales han sido de gran ayuda a la hora de plantear y desarrollar el programa final. Si no se hubiesen realizado previamente, en el momento de implementar el programa final habrían aparecido una amplia cantidad de problemas, desde la necesidad de la realización de una función de coste personalizada, a la transformación de los datos dentro de la misma.

5 Implementación

En este capítulo se va a exponer la implementación del programa final. Se explicarán de forma simultánea el diseño con su posterior desarrollo, de forma que se logre comprender los motivos por los que se realizan los procesos de la forma elegida.

5.1. Introducción

A lo largo de los pasos ejecutados en el desarrollo, se ha comprobado la posibilidad de crear un programa que sea capaz de realizar el cálculo del VaR, mediante un entrenamiento previo. De esta forma, se ha procedido a implementar la unión de las distintas partes necesarias para el correcto funcionamiento del programa.

También se han producido cambios en algunas de las tareas realizadas para la mejora de su funcionamiento, y para adaptarlas a las demás partes que lo componen. Tal y como se comentó en el capítulo anterior, se va a utilizar el diagrama del programa final para la correcta explicación de las distintas partes que lo componen, y del funcionamiento del mismo.

El resultado de las ejecuciones y las pruebas realizadas será comentado en el capítulo de resultados, dónde se explicarán con detalle. También se comentarán allí, los cambios realizados en el programa para cada una de las pruebas realizadas.

5.2. Descripción del programa final

En esta sección se va a explicar la realización del programa final, de forma que se comprendan las decisiones de diseño tomadas finalmente, y la capacidad de la implementación desarrollada. Como se ha comentado, esta explicación se va a basar en el esquema expuesto en el capítulo anterior.

Antes de comenzar a exponer el esquema desarrollado, hay que destacar que el diseño ha sido ideado para que sea posible su implementación con un número arbitrario de activos y de qubits, partiendo como base de una ejecución cuántica con tres qubits y con la misma cantidad de activos correlacionados.

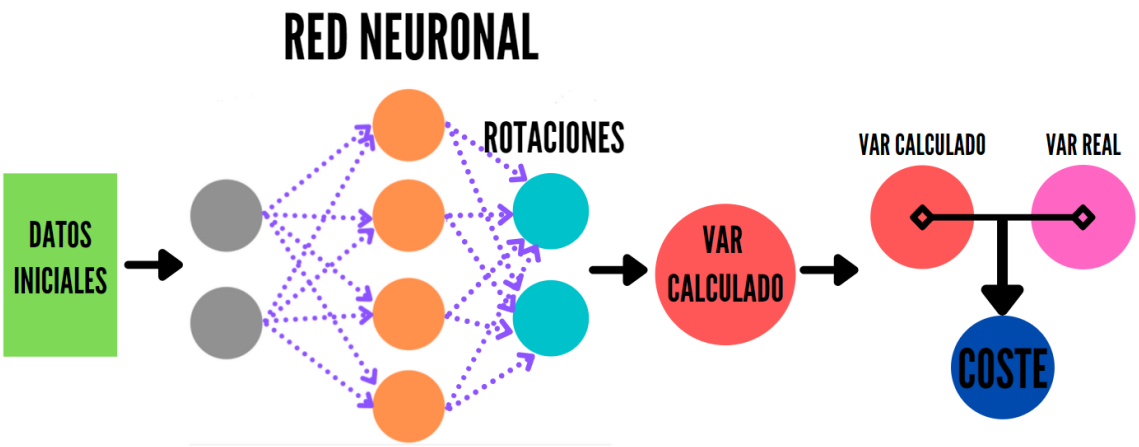

Figura 5.1: Esquema final de la red neuronal con el computador cuántico.

El primer paso de la implementación realizada es la lectura de los datos iniciales suministrados y del VaR histórico obtenido. El cálculo del VaR histórico se realiza previamente en excel, para introducirlo en el programa final como datos iniciales. Este cálculo del VaR histórico se realiza con grupos de 20 en 20 activos, adelantando un día en cada agrupación.

Se ha decidido utilizar estas agrupaciones para generar pares de valores-resultados que habilitan el aprendizaje de la red neuronal. La elección del número exacto se puede variar a deseo del programador cambiando una variable del programa, de forma que se pueden adaptar a distinto número de activos agrupados.

El valor concreto se ha seleccionado después de observar que se pueden obtener valores coherentes del VaR histórico con los datos iniciales, y no genera excesivas variaciones que harían imposible el aprendizaje de los mismos. Hay que destacar que lo datos de los activos iniciales se introducen al programa sin ningún tratamiento previo, ni se ordenan, ni se separan por activos individualmente.

Cuando se realizan los cálculos por el método paramétrico, o de Montecarlo, los activos se multiplican por mil para realizar el cálculo correctamente, sin embargo, en el programa desarrollado este paso tampoco se hace previamente, sino que se espera que la red sea capaz de aprenderlo sola.

Una vez que se han seleccionado los datos iniciales y los distintos VaR asignados a los mismos, estos se introducen en la red neuronal para que sea capaz de aprender de los mismos. Las capas y la cantidad de neuronas son ajustables para el desarrollador, pudiendo variar o pasar directamente a ejecutar el nuevo esquema de la red neuronal. Como activación de las capas se ha seleccionado la activación que proporciona un rango de valores amplio, superior al rango de valores de las posibles rotaciones, pero al ser una rotación, cada 360^{o} se repiten los mismos valores.

Además, se comparará con una activación personalizada, en la que se pueden modificar la variable de variación de los resultados y el valor máximo que puede tener una neurona, pudiendo limitar el rango máximo al rango de las rotaciones exacto.

Con las opciones seleccionadas por el desarrollador, la red neuronal aprende de los datos iniciales y realiza los cálculos necesarios para obtener las rotaciones que se deberían introducir en el circuito cuántico, que arroja el resultado del VaR predicho. Este proceso de aprendizaje para el cálculo de las rotaciones, se realiza a través de la función de coste desarrollada previamente, donde todas las rotaciones calculadas se ejecutan en el sistema cuántico deseado.

El mayor problema encontrado en el momento de la implementación de la función de coste fue

la operativa con los tensores de keras. Estos tensores tienen que mantenerse intactos, variando tan solo los valores con sumas o restas, sin poder ser sustituidos o igualados. Esta imposibilidad encontrada ha fomentado una solución creativa para realizar el paso de las rotaciones calculadas a un solo vector con el valor del VaR calculado en la computadora cuántica.

Todo el desarrollo del programa final se ha llevado a cabo en un entorno local, en vez de utilizarse la plataforma de IBM, como ha ocurrido en la mayoría de las tareas llevada a cabo. Esta decisión se tomó en base a las necesidades que aparecen al utilizar una red neuronal. Principalmente, la razón que hizo que este cambio fuese necesario es la utilización de los módulos típicos de redes neuronales, keras, Tensorflow, etc.

La plataforma de IBM, de la que hay que destacar que ha mejorado de gran forma en los últimos meses, no permite añadir módulos a su programa Jupyter, por lo que, si un módulo no está instalado de forma nativa, no se puede utilizar para las ejecuciones. Por este motivo, el traspaso de la plataforma en la nube a local, era un paso necesario que tenía que ser realizado para poder avanzar con el TFM.

Al realizar el traspaso de la plataforma a local, aparecieron una serie de problemas de programación y conflictos de nomenclatura que necesitaron de arreglos para conseguir que el programa final pudiese funcionar. Una vez solucionados estos problemas se pudo pasar a implementar el esquema final diseñado.

Uno de estos problemas existentes, fue la utilización de una gaussiana aproximada en el momento de extraer los resultados de las ejecuciones. Matemáticamente, se puede demostrar que la suma de una serie de gaussianas se puede modelar como una sola gaussiana aproximada. Si se calculan previamente los valores de la gaussiana aproximada, se puede utilizar los valores de la ventana seleccionada para utilizarse como eje x del cálculo del VaR resultante. De esta forma, al calcular el VaR se puede realizar el cálculo del valor en el que se debe cortar la ventana de forma teórica para obtener el 5 % de la probabilidad a un lado y el 95 % de la probabilidad al otro lado.

Pero este método requería unos cálculos previos, que añadían una necesidad de datos iniciales al programa externo, a los datos de los activos, es decir, indirectamente se estaba realizando un pequeño preprocresado, que, aunque solo afectaba al paso final del extractor de resultados, se consideraba que quizás existía una posibilidad de evitarlo, otorgando la independencia a la red neuronal y al programa, de cualquier análisis o procesado de los datos previo.

El problema que ocurría al utilizar este método era que la ventana de la gaussiana seleccionada no era siempre suficiente para cualquier número de activos, dejando a la red neuronal sin posibilidades de aprender hasta los valores esperados. Debido a esto, se pensó en una forma alternativa de rellenar el eje x de los posibles valores del VaR, adaptando la cantidad de posibles valores, definida según la cantidad de qubits utilizados en el computador cuántico.

En primer lugar, hay que deducir que el VaR resultante, que siempre se va a encontrar dentro del rango que proporciona el mínimo VaR de las agrupaciones realizadas, puesto que, si el VaR es el punto en el que se encuentra el percentil 5 % de la función de distribución de probabilidad, siendo agrupaciones de 20 activos, este valor de las agrupaciones siempre será más "extremo" que el valor final. Una vez se comprende esta deducción, se podrá utilizar a favor del extractor de resultados.

Conociendo los dos extremos en los que se encontrará necesariamente el valor del VaR obtenido, se puede realizar una división de ese mismo rango entre la cantidad de estados posibles que aparecen con los qubits utilizados, dando esto lugar a un posible método que valdría para cualquier cantidad de activos y de valores del VaR posibles. Los resultados de este método se expondrán en el siguiente capítulo detalladamente.

Con los cambios implementados, el flujo de ejecución quedaría de esta forma: al comienzo, los

datos iniciales se agrupan y se almacenan en variables, al igual que los cálculos del VaR histórico parcial de cada agrupación. Una vez se tienen estos datos iniciales almacenados, se pasa a construir el modelo de red neuronal que será utilizado en la ejecución. Después de instanciar el modelo, se comienza el aprendizaje con el número de épocas fijado y la función de coste previamente diseñada (función que utiliza la función creada para personalizar y ejecutar, con el posterior extractor de resultados).

Cuando todas las épocas han finalizado, otorgando la posibilidad de aprender a la red neuronal, se pasa a realizar la validación de los cálculos de la red neuronal. Si este VaR predicho es aproximado al valor deseado, se procede a almacenar los resultados de la prueba para su posterior comparación. Sin embargo, si los resultados están alejados del valor deseado, se pasa a realizar ajustes en las distintas partes del programa para intentar mejorar el funcionamiento del mismo.

Cada uno de estos cambios aporta una nueva perspectiva al problema, haciendo prácticamente imposible encontrar una solución que sea mucho mejor que las demás y otorgando la capacidad de perfeccionar el problema con una nueva forma de afrontarlo. En general, se podría destacar que el programa aporta un beneficio notable a las posibilidades de realizar programas de inteligencia artificial con un computador cuántico, no tanto como programa individual (no por su funcionamiento), sino por las ideas y capacidades que se han conseguido llevar desde la teoría inicial a la práctica final. Por ejemplo, la función de personalización y ejecución podría utilizarse en una amplia variedad de programas, al igual que la función de coste personalizada. De esta forma, se profundiza sobre el conocimiento y las herramientas actuales en ejecuciones cuánticas en conjunto con programas clásicos o redes neuronales.

Por último, hay que destacar las posibilidades de personalización del programa final implementado, que otorgan unas amplias capacidades de mejora y adaptación a diversas situaciones. El programa tiene la posibilidad de cambiar una serie de variables que se exponen en la tabla para mayor claridad.

Variables	**Capacidades**
Qubits	Una variable permite la personalización de la cantidad de qubits de las ejecuciones, y por tanto de los posibles valores que puede resultar el VaR predicho. Esta variable también ajusta automáticamente el número de neuronas de salida de la capa final de la red neuronal.
Proveedor	Con la variable de ejecución real o simulada se puede decidir el método de ejecución, y mediante un token personal que otorga la plataforma de IBM se pueden realizar ejecuciones reales de forma automática, y de forma transparente al desarrollador.
Activos agrupados	Una variable en la lectura de los datos permite realizar agrupaciones por el número que seleccione el usuario, sin problema de seleccionar un valor mayor o menor.
Neuronas	En el modelo de la red neuronal se puede variar la cantidad de neuronas utilizadas en cada una de las capas, permitiendo realizar pruebas con distinta cantidad de neuronas.
Nº de activos	Se puede introducir cualquier cantidad de activos, eligiendo las que serán procesadas con el valor de la variable que selecciona los activos utilizadas.

Cuadro 5.1: Capacidad de personalización del programa.

6
Resultados

En este capítulo se van a exponer las pruebas realizadas y los resultados de los mismos, otorgando de esta forma, una idea de la precisión global de sistema y de las capacidades que proporciona a la hora de realizar el cálculo del VaR con los datos suministrados.

6.1. Introducción

Una vez se ha desarrollado la implementación final, el siguiente paso es realizar una serie de pruebas capaces de validar el funcionamiento y analizar los resultados obtenidos. Para extraer esta serie de resultados se ha decidido realizar distintas pruebas, de forma que se pueda ver el funcionamiento del sistema y mejorarlo con las variables seleccionadas para aumentar la precisión, la velocidad o reducir el rango de posibles resultados.

Como se ha comentado en capítulos anteriores, el resultado que proporciona la ejecución de las rotaciones de la red neuronal, es siempre un resultado discreto, debido a que solo puede tomar los valores indexados en el eje establecido, que en su mayoría será con el método expuesto del rango entre VaR máximo y mínimo. De esta forma, si se pretende aumentar la precisión del sistema, la variable que se debería cambiar teóricamente es la cantidad de qubits establecida, para que existan más estados posibles y se pueda seleccionar uno más cercano al VaR deseado.

Para comprender los resultados de las pruebas ejecutadas, es necesario conocer en primera instancia las bases de datos sobre los que se realizarán estas pruebas. Una vez entendida la base de datos, se puede pasar a observar los distintos resultados obtenidos. Cuando se tienen los resultados, para que estos tengan una mayor validez, es interesante compararlos con un sistema de referencia, otorgando una visión global del problema y de la forma en la que se ha afrontado. Todas estas partes serán explicadas en detalle a continuación.

6.2. Datos utilizados en las pruebas

Para las distintas pruebas realizadas, se han preparado dos bases de datos diferentes, en primer lugar, se realizará unas pruebas sobre una colección de tres activos y, posteriormente, las mismas pruebas sobre una colección de cinco activos. De esta forma se podrá observar el

funcionamiento del programa con una cantidad diferente de activos, pudiendo acreditar que el programa realiza los cálculos con lógica y la precisión real del mismo.

Ambas bases de datos tienen el VaR previamente calculado en las respectivas hojas de excel. De esta forma, se pueden comparar los resultados obtenidos con el VaR paramétrico o de Montecarlo, pudiendo obtenerse conclusiones. Estos resultados del VaR serán expuestos a continuación para mayor claridad.

Por cuestiones de confidencialidad, estos datos no se pueden hacer públicos al tratarse de valores de activos reales, pero sí se expondrán todos los valores estadísticos que puedan llegar a servir para la explicación de los distintos resultados alcanzados.

Por último, hay que destacar que estas bases de datos están formadas por 81 valores de cada acción, habiendo sido tomados estos valores una vez al día, se tiene una base de datos de casi tres meses con la que se intentará calcular el valor del VaR, que se debería utilizar para el riesgo del día siguiente.

6.3. Sistema de referencia

El sistema de referencia, como ya se ha expuesto, será una red neuronal que sea capaz de calcular el VaR directamente, sin necesidad de un computador cuántico para las ejecuciones. Este sistema recibirá exactamente los mismos datos introducidos, sean los tres o los cinco activos. Se agruparán los datos en los mismos paquetes que recibirá el sistema desarrollado.

Este sistema, en vez de arrojar las rotaciones que se deben introducir en el circuito cuántico para obtener el VaR, realiza el cálculo de dentro de la red neuronal y otorga el valor del VaR predicho directamente. El esquema empleado para esta red neuronal será distinto al que se utiliza en el programa que contiene el computador cuántico.

Se intentará optimizar el funcionamiento del sistema para poder comparar los resultados de forma correcta. De esta manera, se probará con distintas combinaciones de número de capas y número de neuronas de cada una de ellas, realizando cambios también en la activación de cada una de las capas.

6.4. Escenarios de pruebas

El primer escenario de pruebas se ha llevado a cabo con los datos de tres activos correlacionados. Esta serie de activos poseen una cantidad de 81 registros diarios, por lo que se poseen datos de tres meses aproximadamente. Teniendo los datos del VaR paramétrico y de Montecarlo calculados sobre los registros de los tres activos, lo que se necesita en este caso es el VaR estimado por el programa desarrollado.

Para la obtención de la estimación se han llevado a cabo una amplia serie de pruebas, variando todos los parámetros de diseño, que facilitaron en el desarrollo del sistema final. Con estos parámetros se ha podido ir ajustando el comportamiento del sistema para obtener una versión final, que es la que mejores resultados aporta para resolver el problema establecido.

Como se comentó en el capítulo anterior, el extractor de resultados del histograma de las probabilidades de las ejecuciones cuánticas, se puede hacer de dos formas distintas. Una forma sería utilizar los valores estadísticos de la gaussiana conjunta aproximada, método que es capaz de obtener resultados en el rango en el que se encuentra el VaR paramétrico (por lo que sería viable), pero no deseado debido a la necesidad del cálculo de la gaussiana aproximada previamente y obtención de características.

Además, este método tiene un problema, que puede impedir el correcto funcionamiento del programa, dicho problema es la selección de la ventana de la gaussiana en la que se encuentra el rango deseado para los resultados. Si la ventana se selecciona incorrectamente, la red neuronal no podrá encontrar un punto cercano (de bajo coste) por más que lo intente. De esta necesidad, surge el método diseñado e implementado de establecer el rango del VaR estimado, utilizando el mínimo y el máximo de las distintas agrupaciones realizadas en el preprocesamiento de los datos.

Sin embargo, aunque se disponga de un rango adecuado para cualquier cantidad de activos o número de agrupaciones, es necesaria otra aclaración, el umbral en el que se selecciona el VaR estimado. Para este cálculo, al tratarse de una ventana de la distribución, no resulta trivial la selección del valor umbral, por lo que se ha decidido realizar distintas pruebas con 5 %, 25 % y 50 %. En estas distintas pruebas se intenta observar el método que arroje mejores resultados para las ejecuciones. Tras una intensa serie de pruebas se ha tomado a la decisión de que el valor adecuado es tomar el 50 % como valor umbral para seleccionar el VaR estimado.

Con la cantidad de qubits se puede aumentar la resolución del eje x del VaR estimado, por lo que, teóricamente, se debería mejorar la precisión del sistema. Este suceso se tratará de comprobar empíricamente. En este caso se han realizado distintas pruebas con 4, 5 y 6 qubits.

También hay que destacar que, dependiendo de la cantidad de capas que compongan la red neuronal, el tipo de las mismas y la cantidad de neuronas que las posean, varía el resultado del cálculo. Esto se ajustará a lo largo de las distintas ejecuciones, para intentar mejorar el sistema con cada decisión de diseño tomada.

Teniendo esto definido, es necesario realizar la elección de las activaciones de las distintas capas. En el caso desarrollado, al necesitar valores con un rango de 2π, se ha seleccionado la activación "relu" y se han realizado pruebas con la función de activación normal y otras con la función con la saturación para que el rango de valores posibles sea el mismo, que el rango de las rotaciones posibles.

Por último, hay que aclarar que se han realizado grupos de 100 pruebas, de forma que se puedan comprobar valores estadísticos como la media de los resultados obtenidos o la moda de los mismos. Observando el comportamiento de forma más precisa.

6.5. Resultados de las pruebas

Una vez se tiene una visión clara del escenario en el que se van a realizar las pruebas, se puede pasar a analizar los resultados de las mismas. Se comenzará exponiendo el resultado de las pruebas realizadas sobre los datos, que contienen, en primer lugar una cantidad de tres activos, y a continuación los de cinco activos.

6.5.1. Pruebas en el simulador

Tres activos

La prueba con los datos de tres activos fue la primera en llevarse a cabo, por lo que los conocimientos adquiridos en la realización de las pruebas han servido para mejorar los resultados del caso de cinco activos. Se van a exponer los resultados del VaR y se realizarán comentarios sobre los mismos.

Mientras que el valor paramétrico tiene un valor exacto, los otros tres métodos pueden otorgar distintos valores del VaR estimado. Para la obtención de los datos del sistema desarrollado, se

ha optado por realizar series de 100 ejecuciones con cinco qubits. Posteriormente, se obtiene la media de los mismos que otorga un valor más preciso, que el de una sola ejecución, debido a la aleatoriedad intrínseca de los circuitos cuánticos. Para una comparación justa se ha optado por realizar lo mismo en la red neuronal, que no utiliza el computador cuántico.

Paramétrico	Montecarlo	Red Neuronal	Red Neuronal (5 Qubits)
857	723-958	877-886	737-921

Cuadro 6.1: Resultados utilizando el simulador con tres activos.

Como se puede observar, el VaR paramétrico es el único que siempre mantiene el mismo valor a lo largo de las ejecuciones. Debido a esto, se va realizar una comparación inicial de los demás métodos con él (aunque luego se va a explicar que beneficios tiene que el valor pueda variar en un rango mayor o menor). En el caso del VaR de Montecarlo, se puede observar que los valores estimados tienden a ser menores al paramétrico, y que posee un rango de 235 de amplitud, se sitúa relativamente cercano al valor paramétrico. Si se observa el valor de la red neuronal que no utiliza el computador cuántico, se puede que apreciar que es el VaR con más similitud al paramétrico, y que además, tiene un rango de posibles resultados mucho menor que los otros dos métodos. Por último, el valor del diseño de la red neuronal, utilizando cinco qubits, arroja valores que pueden ser inferiores o superiores al valor paramétrico y tiene un rango un poco menor que el de Montecarlo (se recuerda que se trata de los valores de la media de 100 ejecuciones del diseño implementado, que otorgan mayor precisión, ya que estos resultados tienen intrínseca la aleatoriedad de los computadores cuánticos).

La causa por la que la comparación con el VaR paramétrico es útil pero no aporta un análisis totalmente correcto, se debe a que el valor paramétrico no tiene en cuenta los posibles sucesos que puedan llegar a ocurrir en el futuro (unicornios y cisnes negros) por lo que otorga un valor estático. Esto sería una buena aproximación si no ocurriese absolutamente ningún suceso que causase una variación no esperada. Debido a que estos suceson ocurren con una amplia frecuencia en este tipo de cálculos, el método que más se utiliza en la actualidad por las entidades financieras es el VaR de Montecarlo, e intentan utilizar el VaR paramétrico en casos en el que el VaR de Montecarlo sea irrealizable por su coste computacional.

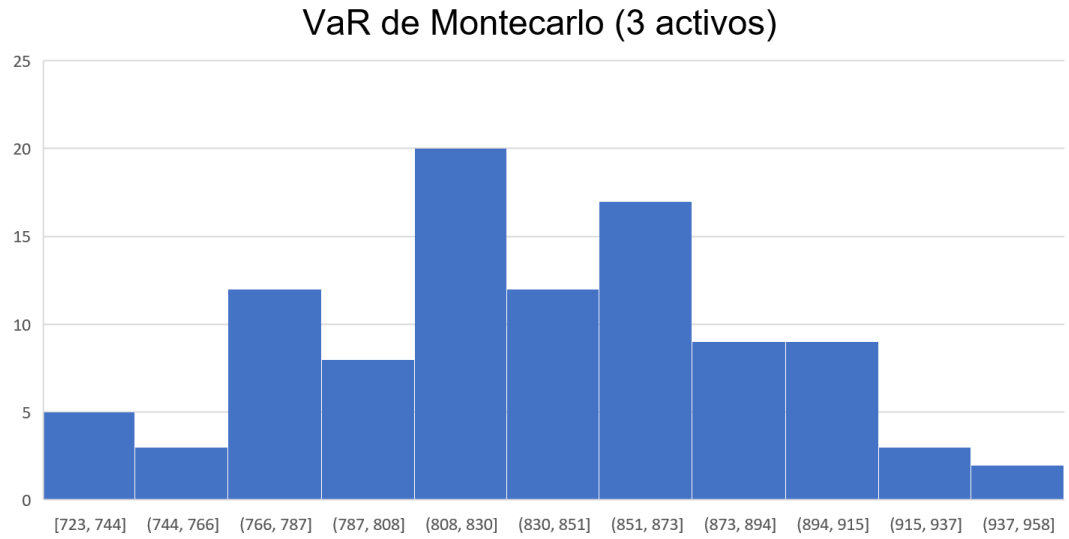

Figura 6.1: Histograma del VaR de Montecarlo (Tres activos).

En esta figura se puede observar el histograma del VaR de Montecarlo para 100 ejecuciones

distintas. Se puede comprobar el rango otorgado previamente y se aprecia que la tendencia es que las probabilidades aumenten hacia el centro de la distribución, exceptuando los casos que serían valores para posibles sucesos que pueden ocurrir. En el caso de que mejore el VaR, se podría dar un unicornio y si empeora se podría dar un cisne negro. Como tiene en cuenta el riesgo que permite prevenir esta estimación de los sucesos, es el método más utilizado en la actualidad para estos cálculos, con el único defecto de su gran coste computacional.

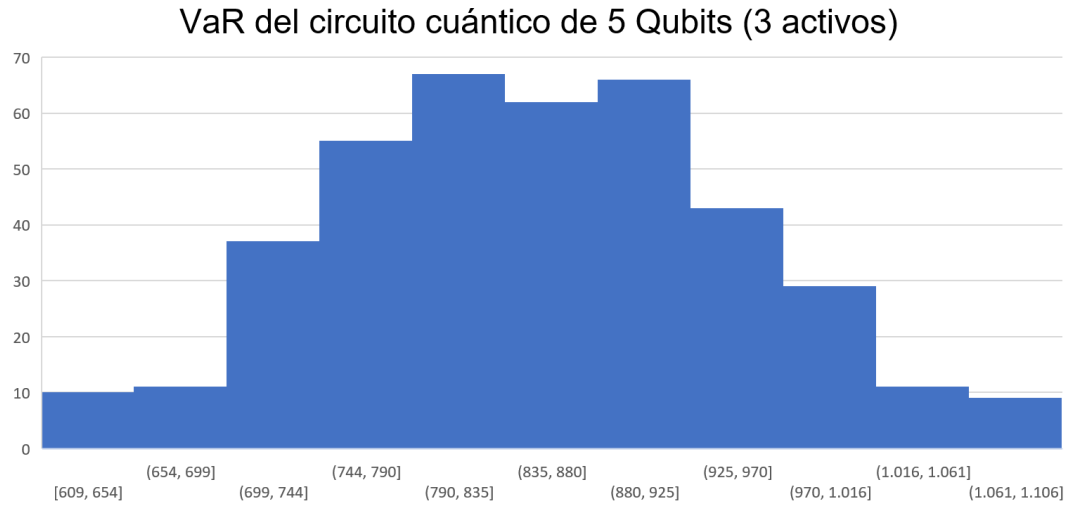

Figura 6.2: Histograma del VaR, de una serie de ejecuciones del que se obtendría la media, de la red neuronal con cinco qubits (Tres activos).

En el caso de la red neuronal, que utiliza el computador cuántico de simulación con cinco qubits, parece que está sucendiendo algo similar al caso de Montecarlo. También se aprecia como las probabilidades van creciendo hacia el centro de la distribución, aunque en este caso, la parte central se encuentra más igualada, por lo que podría indicar que los sucesos tienen una probabilidad equitativa de aparecer. Esto puede indicar que, efectivamente, el sistema está aprendiendo a implementar esta serie de posibles riesgos en los cálculos que realiza a través de los qubits de los circuitos cuánticos, ya que se diferencia en gran medidad de los valores obtenidos por la red neuronal pura (sin computador cuántico).

Cinco activos

Se presentan los datos obtenidos del simulador, y después los resultados obtenidos en ejecuciones reales. Para la obtención de los mismos se ha seguido el mismo proceso que con tres activos.

Paramétrico	Montecarlo	Red Neuronal	Red Neuronal (5 Qubits)
1463	1214-1622	1434-1453	1378-1542

Cuadro 6.2: Resultados utilizando el simulador con cinco activos.

En estas pruebas, también se puede observar, en el caso del VaR de Montecarlo, que los valores estimados tienden a ser menores al paramétrico. Posee un rango de 408 de amplitud, que se sitúa relativamente cercano al valor paramétrico. Si se observa el valor de la red neuronal, que no utiliza el computador cuántico, se puede apreciar, de nuevo, que es el VaR con más similitud al paramétrico, y que además, tiene un rango de posibles resultados mucho menos que los otros dos métodos. Por último, el valor del diseño de la red neuronal utilizando cinco qubits arroja

valores que pueden ser inferiores o superiores al valor paramétrico y tiene, de nuevo, un rango más corto que el método de Montecarlo e incluído dentro del mismo. Por lo que los resultados tienden a ser similares (en cuestión de amplitud y dónde sitúan) que, en el caso con tres activos. Si se observa los valores de los histogramas, se puede observar con más precisión.

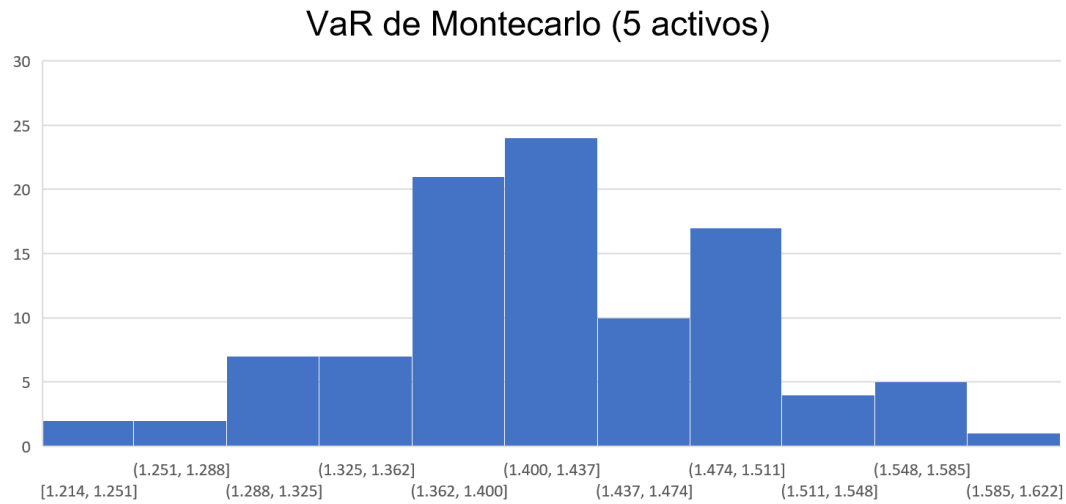

Figura 6.3: Histograma del VaR de Montecarlo (Cinco activos).

En esta figura se puede observar el histograma del VaR de Montecarlo para 100 ejecuciones distintas. Se puede comprobar el rango otorgado previamente y se aprecia que la tendencia es que las probabilidades aumenten hacia el centro de la distribución (en este caso más cláramente), exceptuando los casos que serían valores para posibles sucesos que pueden ocurrir. Por lo que se puede observar una gran similitud al caso con tres activos, pero hay que destacar que , según va creciendo el número de activos utilizado, se produce un incremento del coste computacional, por lo que si se continuase con el incremento de los mismos llegaría un punto (relativamente cercano, 100 por ejemplo) en el que no se pudieses aplicar este método.

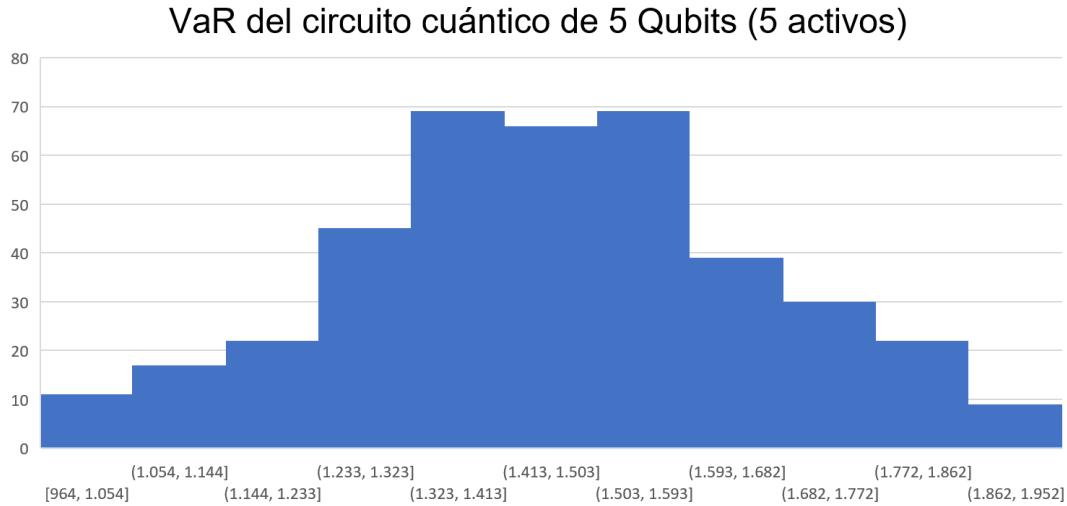

Figura 6.4: Histograma del VaR, de una serie de ejecuciones del que se obtendría la media, de la red neuronal con cinco qubits (Cinco activos).

En el caso de la red neuronal que utiliza el computador cuántico de simulación con cinco qubits, parece que vuelve a suceder algo similar al caso de Montecarlo. También se aprecia como las probabilidades van creciendo hacia el centro de la distribución. Esto puede indicar que,

efectivamente, el sistema sí estaría aprendiendo a implementar esta serie de posibles riesgos en los cálculos que realiza a través de los qubits de los circuitos cuánticos. Por lo que se podría considerar un método que incluiría los riesgos en los cálculos con un coste computacional que casi no crecería por cada nuevo activo incluído.

6.5.2. Pruebas en el computador cuántico

Por último, se han realizado pruebas reales sobre los cinco activos. Para la ejecución se ha seleccionado el computador cuántico de Santiago de Compostela de la plataforma de IBM. Debido al modelo de ejecución por posición en la cola, se han podido ejecutar un número inferior de pruebas a las simuladas. Por lo que se aporta directamente el histograma obtenido, con el resultado estimado señalado en la propia gráfica.

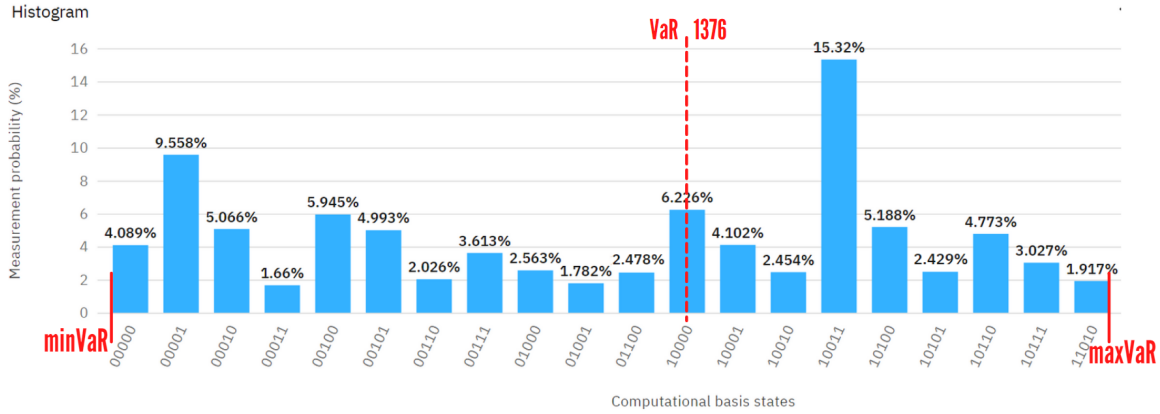

Figura 6.5: Resultado final de la estimación del VaR en el computador cuántico real.

En el caso de la ejecución del diseño en un sistema real, se puede observar que el diseño sigue funcionando (aunque de forma ligéramente peor) incluso ejecutándose en un computador cuántico real. Aunque este paso del simulador al computador real parezca un paso trivial, resulta ser el punto en el que la amplia mayoría de diseños tienen más problemas, debido al ruido que añade la ejecución real. Debido a esto, se puede comprender que la red neuronal estaría aprendiendo a modelar este ruido como una vareable más de diseño, pudiendo reducir los efectos de los mismos en el cálculo del VaR. De esta forma, se estaría habilitando el diseño y la obtención de resultados de un computador cuántico real, proceso que aceleraría en gran medida el cálculo del VaR de este nuevo método de obtención planteado.

Se observa que el histograma que aporta la plataforma de IBM en ejecuciones reales de cinco qubits está incompleto, pues este no suma exactamente 100 %. Esto se debe a la selección de los estados con las 20 mayores probabilidades, o a la eliminación de los estados que solo aporten ruido (por ejemplo, primero ejecutando el circuito simulado y eliminando los estados que no aparecen en las simulaciones).

Con esto, el resultado obtenido con la totalidad de los estados mejoraría, puesto que desplazar la división del VaR hacia la izquierda significa elevar el valor estimado, con lo que se aproxima más al valor paramétrico. Esta puede ser la razón por la que se ha observado, que los valores obtenidos en las simulaciones reales tienen un rango de valores menor, pudiendo variar entre 1271-1441. Si estos valores "descartados" por su menor probabilidad no pudiesen incluirse de otra forma, se podría variar el valor de la probabilidad con la que se estima el VaR al 40 %, y con esto, podrían mejorar los valores obtenidos.

7

Conclusiones y trabajo futuro

7.1. Conclusiones

A lo largo de la realización de este Trabajo Fin de Máster, se ha adquirido una gran cantidad de conocimiento sobre el diseño de los circuitos cuánticos y la implementación de la computación cuántica con redes neuronales.

Se ha comprobado la posibilidad de implementar el diseño de las rotaciones de los circuitos cuánticos con redes neuronales, forzándolas a aprender sobre los mismos mediante funciones de coste personalizadas. También se ha observado las amplias posibilidades que otorgan las librerías de Python utilizadas, con las que se pueden desarrollar funciones para editar circuitos cuánticos, extraer los resultados de los histogramas o crear funciones capaces de realizar cálculos en computadores clásicos y cuánticos.

Además, durante el desarrollo, se ha podido observar las capacidades que poseen los computadores cuánticos, los qubits que suelen poseer y los efectos que aporta el aumento los mismos en los cálculos que se realizan. Mientras se realizaban los pasos y las ejecuciones, se adquirieron conocimientos sobre las posibilidades que aporta la plataforma de desarrollo cuántico de IBM, donde se pueden crear circuitos, programar y realizar ejecuciones visualizando los resultados.

En el ámbito financiero, se ha podido comprobar la posibilidad de desarrollar una red neuronal (sin computador cuántico), que sea capaz de obtener una estimación del VaR que se acerca a los valores predichos por los métodos clásicos, reduciendo el coste de ejecución de forma incremental según se aumenta el número de activos.

A este objetivo realizado, le ha seguido el que era el objetivo principal de este TFM, lograr desarrollar un diseño que sea capaz de realizar estimaciones del VaR, utilizando circuitos cuánticos personalizados con las rotaciones que aporta la red neuronal. De esta forma, se ha obtenido una implementación que es capaz de aprender de los datos de los activos introducidos y realizar una estimación del VaR, que, usualmente, resulta ser más precisa que la obtenida con el sistema que no implementa el diseño de circuitos cuánticos (en base a las pruebas realizadas).

Por último, se ha podido observar la posibilidad de que las redes neuronales puedan aprender sobre el sistema en el que se están ejecutando. Esto posibilita que, consigan reducir el ruido introducido por los sistemas cuánticos reales. Esta unión podría llegar a ser muy beneficiosa,

en el caso de que la implementación de una red neuronal con un computador cuántico, consiga aprender a modelar este ruido, como una variable aleatoria más, permitiendo de esta forma realizar cálculos con mayor precisión.

Con todos estos puntos expuestos, se puede corroborar que se han cumplido los objetivos propuestos en este TFM, adquiriendo un conocimiento sobre las redes neuronales en conjunto con el diseño de circuitos cuánticos, que podría ser valioso para trabajos futuros o, posibles investigaciones o implementaciones, que tomen algunas de las ideas expuestas en este Trabajo Fin de Máster como base o referencia.

7.2. Trabajo futuro

Con base en este Trabajo Fin de Máster, se podrían realizar algunas mejoras o nuevas implementaciones como trabajo futuro, de forma que el conocimiento adquirido a lo largo de su realización, permita continuar con el proceso de investigación en el campo de la computación cuántica, las redes neuronales, e incluso, los riesgos financieros. Pudiendo suponer un punto de partida para el perfeccionamiento de este posible método de estimación del VaR de los activos de las entidades financieras.

Se aportan algunas de las posibles ideas que pueden realizarse como continuación de este TFM, pudiendo ser realizados en otros TFMs que continúen esta o nuevas investigaciones.

- Implementación del esquema de aprendizaje automático GAN (*Generative Adversarial Network*): Este tipo de implementación de Machine Learning ha surgido a lo largo de los últimos años. Se basa en el aprendizaje mediante dos redes neuronales distintas, actuando una como calificadora de los cálculos que realiza la otra. Con este tipo de implementaciones se puede mejorar el rendimiento del aprendizaje, de forma que la red neuronal resultante funcione mejor que las que reciben un entrenamiento normal.
- Desarrollo de un sistema de inteligencia artificial, que sea capaz de diseñar circuitos cuánticos genéricos según las necesidades del problema (cantidad de puertas lógicas, tipos, etc), que posteriormente puedan ser personalizados cambiando sus rotaciones. De esta forma se podrían necesitar varias redes neuronales que vayan cumplimentando las necesidades de las demás.
- Implementación de una red neuronal que sea capaz de disminuir el efecto del ruido cuántico, de forma que se puedan desarrollar diseños cuánticos en el simulador y que los resultados en la ejecución real sean similares. Si se consiguiese este desarrollo la facilidad para investigar sobre la computación cuántica se incrementaría de forma significativa.

Glosario de acrónimos

- **VaR**: Valor a riesgo
- **TFM**: Trabajo Fin de Máster
- **IA**: Inteligencia Artificial
- **QASM**: Quantum Assembly Language
- **MIT**: Massachusetts Institute of Technology
- **GAN**: Generative adversarial network

Bibliografía

[1] Steane, A. (1998). Quantum computing. Reports on Progress in Physics, 61(2), 117.

[2] Preskill, J. (2018). Quantum Computing in the NISQ era and beyond. Quantum, 2, 79.

[3] Alpaydin, E. (2020). Introduction to machine learning. MIT press.

[4] Lov Grover and Terry Rudolph (2008), Çreating superpositions that correspond to efficiently integrable probability distributions", arXiv:quant-ph/0208112v1 15 Aug 2002

[5] Glasserman, P. (2003). Monte Carlo Methods in Financial Engineering. Springer-Verlag, New York.

[6] PAGÈS, G. (2018), "Numerical Probability: An Introduction with Applications to Finance", Ed. Springe.

[7] Stefan Woerner and Daniel J. Egger (2018), "Quantum Risk Analysis", arXiv:1806.06893v1 [quant-ph] 18 Jun 2018

[8] Mikko Mottonen,Juha J. Vartiainen, Ville Bergholm, and Martti M. Salomaa (2008), "Transformation of quantum states using uniformly controlled rotations", Xiv:quant-ph/0407010v1 1 Jul 2004

[9] Anthony, M., Bartlett, P. L. (2009). Neural network learning: Theoretical foundations. cambridge university press.

[10] Basheer, I. A., Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. Journal of microbiological methods, 43(1), 3-31.

[11] Open-Source Quantum Development. (n.d.). Retrieved February 07, 2021, from https://qiskit.org/

[12] Daniel Koch, Laura Wessing, Paul M. Alsing (2019), Ïntroduction to Coding Quantum Algorithms: A Tutorial Series Using Qiskit", arXiv:1903.04359v1 [quant-ph] 7 Mar 2019.

[13] Knill, E. (2005). Quantum computing with realistically noisy devices. Nature, 434(7029), 39-44.

[14] StephanJ98, “Qubit, El Secreto De Los Ordenadores Cuánticos.”, Scripters, 17 Oct. 2019, scripters.es/qubit-el-secreto-de-los-ordenadores-cuanticos/.

[15] IBM, IBM Quantum Experience. Retrieved February 07, 2021, from https://quantum-computing.ibm.com/

[16] Wilson, T. (1999) “Value at risk” in Risk Management and Analysis, Vol. 1 pp. 61–124, C. Alexander, ed., Wiley, Chichester, England.

[17] Vergels, Amparo. “VaR Histórico - Definición, Qué Es y Concepto.” Economipedia, 4 Feb. 2020, economipedia.com/definiciones/var-historico.html.